UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Desarrollo de aplicación móvil para terminales industriales

Fernando Carrascosa García

Tutor: David González López
Ponente: Gustavo Sutter Capristo

Junio 2015

Agradecimientos

A mi familia.

A mis compañeros y amigos. Los que quedaron atrás y los que llegan a la meta.

A los que tienen ilusión por enseñar.

Resumen

El presente Trabajo Fin de Grado trata sobre el diseño, desarrollo y prueba de la aplicación Django MSM v2.0. La implementación de dicha aplicación se llevó a cabo en la empresa de desarrollo software Athelia Solutions y el cliente al que va destinada es la empresa cervecera Mahou San Miguel.

Django MSM v2.0 es una aplicación que permite gestionar operaciones de mantenimiento sobre instalaciones de cerveza. Estas operaciones de mantenimiento se llevan a cabo por parte de técnicos subcontratados por MSM y las instalaciones de cerveza pertenecen a clientes que contratan los servicios de la empresa cervecera.

MSM incorpora en sus instalaciones de cerveza un sistema de trazabilidad de activos (gestionado por Athelia) que se basa en la identificación de un material de la instalación por medio de una etiqueta electrónica (RFID). Esta etiqueta debe ser identificada, con un lector integrado en un terminal móvil industrial, en todas las operaciones que realizan los técnicos. De esta forma se ponen a disposición del operario todos los datos de la instalación. El uso de la tecnología RFID permite: identificar unívocamente los activos de las instalaciones in situ, agilizar los procesos de mantenimiento y asegurar la presencia de los técnicos durante dichos procesos.

La aplicación desarrollada para este TFG, sustituirá a una versión previa de la misma que requería una renovación total debido a: la descatalogación de los terminales utilizados por el cliente en ese momento, la renovación total del parque de etiquetas RFID (tags) en los activos y la inclusión de nuevas funcionalidades. El objetivo de la renovación es proporcionar a MSM un software más fiable y la capacidad para adaptarse a un mercado de terminales móviles industriales que evoluciona rápidamente.

Palabras clave

Terminal móvil industrial, trazabilidad de activos, tag, RFID y mantenimiento de instalaciones de cerveza.

Abstract

This End-Of-Grade paper is about the design, development and testing of the Django MSM v2.0 application. The implementation of this software was carried out in the software development company Athelia Solutions and the target client is the beer company Mahou San Miguel.

Django MSM v2.0 is an application which helps to manage the maintenance operations in beer installations. These maintenance tasks are conducted by MSM technicians and the beer installations belong to customers that hire the services of the beer company.

The MSM beer installations include an asset traceability system (produced by Athelia), which is based on the asset identification by an electronic tag (RFID). This tag should be identified by a reader, integrated in an industrial handheld device, in all the operations performed by the technicians. By this way, all the installation data will be easily available for the operator. The RFID identification system allows to: univocally identify the assets in the customer installations, accelerate the maintenance processes and guarantee the presence of the technicians during these processes.

The application is going to replace a previous software version and requires a total upgrade due to: the obsolescence of the handheld device in use currently, the upgrade of all tags and the addition of new functions requested by the client. The goal is to provide reliable software to MSM and to continue the adaptation to the changing industrial mobile world, which evolves constantly.

Keywords

Industrial handheld device, assets traceability, tag, RFID and maintenance of beer installations.

Índice de contenido

Índice de tablas

Índice de ilustraciones

Glosario

Lista de términos y abreviaturas que serán utilizadas a lo largo del documento:

Término/ Abreviatura	Descripción
MSM	**Grupo Mahou San Miguel**
HHC	**Hand Held Computer**: Ordenador de mano con Sistema Operativo Windows Mobile.
ISO	**International Organization for Standardization**
DAMM	**Grupo DAMM**
WEHH	**Windows Embedded Handheld**
SVN	**Apache Subversion:** Herramienta de control de versión open source. Se basa en un sistema de ficheros o repositorio. Implementa el sistema de revisión, que sólo guarda el conjunto de modificaciones entre dos archivos, optimizando el espacio en disco.
VS	**Microsoft Visual Studio (2008)**
Tag	Etiqueta electrónica basada en tecnología RFID
RFID	**Radio Frequency IDentification** (Identificación por radio frecuencia).

1 Introducción

En Agosto de 2014 la Escuela Politécnica Superior me ofreció la posibilidad de realizar prácticas en empresa como parte de mi formación académica. Fui destinado a una mediana empresa de la zona industrial de la ciudad de Tres Cantos llamada Athelia Solutions. Después de una excelente acogida y unas semanas de adaptación, comencé a trabajar en varios proyecto de aplicaciones móviles en su mayoría basados en C# .NET y Windows Mobile 6.0.

Terminado el periodo de prácticas se me ofreció la posibilidad de ampliar dicho periodo para permanecer en la empresa hasta verano de 2015. Además, dos proyectos interesantes estaban en marcha, uno de ellos era la renovación de la aplicación Mahou San Miguel. Dadas las circunstancias decidí que era una buena oportunidad para realizar mi Trabajo Fin de Grado en Athelia y compatibilizar así la vida académica con la laboral. No fue fácil, pero en mi opinión, el realizar un proyecto como este en un entorno laboral real, es una gran experiencia.

Entre los productos destacados de Athelia se encuentra Django, un framework para el desarrollo de aplicaciones móviles en el campo del mantenimiento industrial, que constituye la base de distintas aplicaciones desarrolladas en la empresa, entre ellas, la descrita en este TFG. Sobre este software se hablará ampliamente en la sección 3.2.

Django MSM v1.0 nació en el año 2009, fruto del acuerdo entre Athelia Solutions y Mahou San Miguel. Athelia Solutions desarrollaría una aplicación para gestionar de manera eficiente las operaciones que realizan los técnicos de MSM en las instalaciones de cerveza de los clientes. Aunque estos conceptos resulten algo difíciles de entender en este punto del documento, se profundizará en ellos en secciones siguientes.

Por tanto, el contexto general es, por un lado, la industria cervecera, en especial el negocio referente a las instalaciones que sirven cerveza en establecimientos, y por otro lado, la industria de la trazabilidad de activos que facilita la identificación de materiales de un cierto valor o interés. No se debe olvidar que se trata de un entorno de carácter industrial, con terminales más toscos que los que se acostumbran a ver en un entorno de público general. El desarrollo de aplicaciones en este tipo de negocios tiene una dificultad añadida que es la de evitar cualquier error por pequeño que sea, dada la delicadeza del conjunto.

El proyecto Django MSM v1.0 permanece activo en el momento de escribir estas líneas, aunque sería más correcto señalar que la versión actual de la aplicación es la 1.2 (se mantendrá la notación 1.0 para facilitar la lectura). El número de versión da una idea de las muchas modificaciones y puntuales correcciones que ha experimentado el software desde su concepción.

En el año 2014, MSM, después de utilizar la aplicación durante 5 años, se planteó la necesidad de realizar una renovación tecnológica para adaptarse a las posibilidades que ofrecían terminales más modernos y potentes. Esta adaptación pretendía la inclusión de nuevas funcionalidades que suponían cambios relevantes tanto en la aplicación móvil cómo en el lado del servidor. En definitiva, significaba la modernización completa del sistema de mantenimiento.

Entre tanto, el software Django también había avanzado con el paso del tiempo. Las posibilidades que ahora ofrecía eran superiores a las de la versión que utilizaba el proyecto MSM v1.0. Por eso, una renovación del proyecto implicaba forzosamente la utilización de la versión más actual de Django.

Tras un estudio detallado, Athelia llegó a la conclusión de que la mejor opción era realizar una aplicación nueva conservando las funcionalidades del proyecto inicial pero adaptado al nuevo núcleo Django.

1.1 Motivación

La motivación personal para este TFG era aprovechar la oportunidad de realizar el trabajo en un entorno profesional e ir conociendo más a fondo la forma de trabajar en una empresa de desarrollo de software.

Tal y como se expondrá a lo largo del documento, la motivación para las compañías implicadas no era otra que la de aumentar su volumen de negocio. Mahou San Miguel aprovecharía las ventajas que le ofrecía contar con una aplicación modernizada para gestión de mantenimiento, adaptada a nuevos terminales y con funcionalidades nuevas respecto a la anterior. Athelia Solutions conseguiría un nuevo contrato con la empresa cervecera y además, desde la perspectiva de los desarrolladores, la modernización de la aplicación favorecería el mantenimiento futuro de la misma al ser adaptada a la versión más actual de Django.

1.2 Objetivos

El objetivo de este TFG es la creación de una aplicación desplegada en terminales móviles industriales para gestionar las operaciones de mantenimiento realizadas por técnicos de MSM en las instalaciones de venta de cerveza de sus clientes.

Se persigue crear una aplicación fiable, adaptada a las necesidades del cliente, a las nuevas tecnologías y que integre la versión actual de Django, lo que supondrá una mejora de la calidad del software y facilitará su mantenibilidad futura.

La nueva aplicación conservará las funcionalidades de la aplicación anterior y añadirá otras nuevas. Las funcionalidades principales de la aplicación son realizar operaciones de mantenimiento en instalaciones (montajes y desmontajes de instalaciones, limpiezas, reparaciones, etc.)

1.3 Estructura de la memoria

La memoria queda dividida en seis apartados que se detallan a continuación:

- **Estudio del estado del arte:** En este apartado se tratará de dar una visión detallada del contexto en el que se sitúa el desarrollo de la aplicación. También se expondrán las principales herramientas y tecnologías utilizadas para completar el proyecto.

- **Diseño**: En este apartado se tratarán todos los aspectos relacionados con el diseño del proyecto. Se analizará cómo es el proceso que debe recorrer el software desde que se plantea hasta que finalmente se desarrolla. Será introducido el entorno Django sobre el que se asienta el proyecto, analizando sus características principales. Las secciones dedicadas al diseño del proyecto, incluirán explicaciones sobre su funcionalidad, detalles técnicos e irán acompañadas de diagramas explicativos para facilitar la compresión de lo que se expone.

- **Desarrollo e implementación**: En este apartado se tratarán cuestiones relacionadas con el desarrollo de la aplicación. Estas incluyen la creación de la estructura general del proyecto en el entorno de desarrollo, su integración con las herramientas de control de versiones y otros detalles concretos sobre el trabajo realizado. Aparecerán junto a las explicaciones, pequeños fragmentos del código desarrollado.

- **Pruebas**: En este apartado se expondrán las distintas pruebas a las que fue sometido el software. En primer lugar los test unitarios, para pasar luego a los test de integración. Los primeros se centran en módulos independientes de la aplicación mientras que los segundos prueban el funcionamiento conjunto de los mismos.

- **Conclusiones**: En este apartado se resumirán las ideas clave extraídas después de completar el trabajo y se propondrán acciones para el futuro.

- **Anexos**: En este apartado se encuentran elementos que no encajan en el cuerpo principal del documento debido a su extensión o temática, por ejemplo, las tablas de planificación temporal utilizadas.

2 Estudio del estado del arte

En los siguientes apartados se ofrece al lector un estudio detallado sobre aspectos clave para la compresión del trabajo realizado. Se analizan las empresas protagonistas en el proyecto, Athelia Solutions y Mahou San Miguel, exponiendo sus principales áreas de negocio, la trazabilidad de activos y el negocio de la cerveza respectivamente.

Se ofrece una visión general acerca del entorno de desarrollo y las herramientas utilizadas. También se incluyen explicaciones acerca de los modelos de terminales sobre los que se despliega la aplicación y el SO presente en ellos.

2.1 Athelia Solutions

Athelia Solutions (1) es una compañía líder en trazabilidad de activos industriales mediante dispositivos basados en RFID. Athelia Solutions pertenece al grupo AirLiquide concretamente a la división de servicios (Air Liquide Services Division).

Athelia Solutions está formada por un grupo de profesionales enfocados claramente a un ámbito industrial. Este entorno requiere una atención especial hacia la fiabilidad y calidad del producto.

La empresa cuenta con dos sedes situadas en Madrid y París. La comunicación entre ambas es constante ya que determinados proyectos se desarrollan conjuntamente. Esto aporta un interesante aspecto internacional al trabajo que se realiza en la compañía. El grupo está formado por alrededor de 300 profesionales

Entre los productos más destacados de la empresa se encuentra “Django”, un framework de desarrollo para aplicaciones móviles que será las bases del proyecto realizado para este TFG.

El enfoque industrial, la búsqueda de la excelencia y el compromiso con el cliente hacen de Athelia una empresa única en el sector.

Ilustración 1: Logo Athelia Solutions

2.2 La trazabilidad de activos

La trazabilidad es la capacidad para seguir el historial de algo que está bajo consideración. Para garantizarla es necesario adoptar un sistema efectivo de identificación, y es ahí donde entra la tecnología RFID.

La trazabilidad de activos es un campo sorprendentemente amplio y muy presente en casi todas las industrias, en especial aquellas que manejan productos envasados o protegidos. La trazabilidad permite la identificación unívoca de un producto o la construcción de una instalación asociada a un activo.

Algunos ejemplos de activos que precisan trazabilidad son: cilindros de gas (butano, oxígeno medicinal, etc.), barriles de líquidos (cerveza, combustibles, etc.), palés, contenedores, enfriadores y tanques en instalaciones de cerveza, etc.

Ilustración 2: Activos trazables

Este trabajo se centrará en la trazabilidad de activos basada en dispositivos RFID (2). El sistema nace alrededor de los años setenta (3), aunque ha evolucionado mucho con el paso del tiempo. Los protocolos de comunicación que maneja se encuentran actualmente estandarizados por ISO. Tanto los fabricantes de chips como los fabricantes de lectores deben adaptarse a estos estándares.

La trazabilidad basada en RFID requiere de dos elementos principales:

- **El elemento identificador o etiqueta:**

  Recibe el nombre de “etiqueta”, “tag”, “Identificador electrónico”, etc. Es el dispositivo que contiene la información identificadora. Normalmente, se trata de un código alfa-numérico único de longitud variable. El elemento identificador se fija físicamente al objeto que se quiere identificar (se adhiere, remacha, suelda…)

Ilustración 3: Tags variados

- **El elemento lector:**

Es el dispositivo encargado de recibir la información del elemento identificador y procesarla. En este caso, los lectores se encuentran integrados en lo que se denomina "terminales móviles industriales", dispositivos que además ejecutan las aplicaciones que se desarrollan.

Estos terminales son de características muy variadas y existen diversas marcas de este tipo de productos. Normalmente, están adaptados a entornos industriales que implican un desgaste mayor y un uso continuo del aparato, de ahí que resulten algo toscos en comparación con lo dispositivos portátiles destinados a un público más general. En muchas ocasiones también deben adaptarse a entornos de riesgo como atmósferas explosivas.

2.3 Terminales móviles industriales

En este apartado, se presentan brevemente los dos modelos de terminales móviles industriales que han sido utilizados durante el desarrollo del proyecto. El primero, denominado WAP3 para abreviar, es el terminal utilizado para la versión 1.0 de la aplicación, mientas que el segundo, denominado BLACK, es el terminal sobre el que se despliega la nueva versión de la aplicación. Ambos terminales pueden conectarse a un PC a través de una "cuna" de carga. Dichas cunas no se incluyen normalmente al adquirir los terminales.

2.3.1 Psion Teklogix WorkAbout Pro3

El WAP3 es un terminal preparado para entornos industriales desarrollado por la compañía Psion (actualmente Motorola-Zebra). Contaba con un amplio catálogo de elementos de expansión para adaptar el terminal a las necesidades del cliente.

Técnicamente se concibió como un terminal para trabajos basados en movilidad (4), su forma ergonómica y teclado permitían trabajar al operario con una sola mano. Lo cierto es que su gran tamaño y peso lo hacen poco manejable. Actualmente se encuentra descatalogado y no cuenta con soporte técnico.

Sus especificaciones base son:

- Procesador PXA270 624MHz
- 1GB Flash ROM, 256 MB RAM
- Sistema Operativo: Microsoft® Windows® Mobile 6
- Bluetooth® y WIFI integrado.
- Pantalla táctil resistiva de 3.7 pulgadas, full VGA 480x640
- Dimensiones físicas (mm): 223 x 100 x 42
- Ranuras para tarjeta SD y SIM (GPRS compatible).
- Batería estándar: 3.7V, 3300 mAh high-capacity battery.

Algunos módulos de expansión:

- Lector de código de barras.
- Módulo RFID, con módulos HF, MIFARE, LF y UHF.
- Batería alta capacidad: 3.7V, 4400 mAh superhigh-capacity.

Ilustración 4: Psion Teklogix WorkAbout Pro3

2.3.2 Honeywell Dolphin Black 70e

El BLACK es un terminal tipo PDA industrial que combina un atractivo y ligero diseño con una durabilidad muy respetable. Además, sus características hardware lo hacen uno de los terminales más rápidos y eficientes del mercado actual. Como es natural, Honeywell está bastante orgulloso de su creación y asegura que el terminal cuenta con cristal resistente a roturas (Corning® Gorilla Glass®), una pantalla táctil capacitiva con retroiluminación y potentes sistemas para asegurar la conectividad wireless (5).

Sus especificaciones base son:

- Procesador TI OMAP 1GHz
- 1GB Flash ROM, 512 MB RAM
- Sistema Operativo: Microsoft® Windows® Embedded Handheld 6.5
- Bluetooth® v2.1.
- Micro usb conector. Headphone/mic combo Jack.

- WLAN IEEE 802.11 a/b/g/n; Wi-FiTM certified
- Sensores: Acelerómetro, vibración, luz ambiental y proximidad.
- Pantalla táctil capacitiva de 4.3 pulgadas, WVGA 480x800, 2 finger touch.
- Dimensiones físicas (mm): 134 x 73 x 18
- Ranuras para tarjeta microSD y SIM (GPRS compatible).
- Lector NFC integrado.
- Cámara 5.0 megapixel, autofocus, flash. Integra bar code scanning software.
- Batería estándar: Li-ion 3.7 V, 1670 mAh

Algunos módulos de expansión:

- Batería alta capacidad: Li-ion 3.7 V, 3340 mAh.

Ilustración 5: Honeywell Dolphin Black 70e

2.4 Mahou San Miguel

Mahou San Miguel (MSM) es la compañía cervecera líder en España con capital enteramente español. Se trata de la cervecera con más impacto internacional de nuestro país produciendo el 75% de la cerveza española que se bebe en el mundo.

La compañía nace en el año 2000, cuando Mahou, empresa fundada en 1890, adquiere San Miguel. En 2004, se incorpora la marca canaria Reina y en 2007 Cervezas Alhambra. En 2011 se une al grupo Solán de Cabras diversificando el ámbito de negocio (6).

La empresa cuenta con alrededor de 2600 profesionales, 10 centros de producción en total: 8 repartidos por toda la Península Ibérica, 1 en Tenerife y 1 en Rajasthan, India. Cuenta con dos oficinas centrales situadas en Madrid y Barcelona.

Entre los productos de la empresa pueden destacarse “Mahou Cinco Estrellas”, “San Miguel 0,0”, “Mixta”, etc. La presentación de los productos incluye latas de aluminio, botellas o botellines de vidrio y barriles de diferentes capacidades.

2.4.1 Proveedores MSM

MSM cuenta con un servicio de proveedores encargados de suministrar productos a comercios bajo su marca corporativa. Normalmente, estos establecimientos que solicitan los productos reciben ciertas ventajas por la compra, ya que entran a formar parte de un denso entramado comercial.

MSM, al recibir las peticiones de compra, realiza el montaje de instalaciones que sirvan sus productos en el establecimiento solicitante. Estas instalaciones se ceden en régimen de alquiler, lo que significa que el equipo instalado sigue siendo propiedad de Mahou San Miguel.

Normalmente, este servicio está subcontratado a particulares externos a MSM y que realizan las labores “de campo” visitando los distintos establecimientos que les son asignados. Además de los trabajos de montaje, deben cumplir otra serie de tareas cómo el mantenimiento periódico de la instalación (limpieza, por ejemplo) y la resolución de fallos imprevistos, entre otras.

Para organizar el trabajo de los técnicos se forman grupos de acción según la localización de los establecimientos. Se puede decir que el mapa de clientes se divide y reparte entre las subcontratas para optimizar los desplazamientos. El concepto de grupo es importante ya que según el grupo al que pertenezca el técnico, este tendrá acceso a los datos de unas instalaciones o de otras. En el apartado de diseño de la aplicación se amplían los datos sobre las subcontratas, los grupos y los técnicos.

2.4.2 Instalaciones MSM

Se entiende por instalación de cerveza al conjunto de elementos presentes en un establecimiento y que permiten la venta de cerveza en sus diversas presentaciones ofreciendo una garantía de calidad óptima.

Las instalaciones de cerveza cuentan con una serie de elementos comunes sea cual sea la marca distribuidora del producto (7). Estos elementos son:

1. **Botella de CO_2:** Cilindro metálico que contiene el gas CO_2 a alta presión (60-80 bar). Introducirá presión al barril para permitir la circulación de la cerveza. Incluye una llave de salida y una válvula de seguridad. Se conecta a un manorreductor que regula el flujo de salida del gas.

2. **Manorreductor:** Dispositivo para regular la presión de salida del gas CO_2. Está compuesto de una llave de paso y dos manómetros. Los manómetros indican la presión de entrada y de salida del gas.

3. **Barril (Keg):** Contenedor metálico generalmente de acero inoxidable. Su capacidad estándar es de 50 litros siguiendo una regulación europea, aunque existen otros tamaños. Presenta un orificio con dos válvulas automáticas, una para extraer la cerveza y otra para inyectar gas a presión.

4. **Cabezal acoplador:** Dispositivo que permite extraer cerveza del barril e inyectar gas a presión al mismo tiempo. El proceso de acoplar el cabezal al barril se denomina "*pinchar el barril*".

5. **Enfriador:** Dispositivo que enfría la cerveza antes de ser servida. Contiene dos serpentines por los que circulan gas refrigerante y cerveza respectivamente. Ambos se encuentran sumergidos en una cuba con agua que es agitada por un sencillo motor. El agua se enfría con el serpentín que contiene el gas y a su vez enfría el serpentín de la cerveza. Es un elemento aparentemente sencillo pero de los más caros de la instalación, por ello suele ser el elemento al que se le acopla el tag identificador de la misma.

6. **Bandeja:** Lamina de acero inoxidable sobre la que se asienta la columna. Suele incluir un escurridor para recoger la cerveza que se pierde inevitablemente al servirla.

7. **Columna:** Se denomina columna al cilindro que canaliza el circuito de cerveza hasta el grifo. Suele ser metálico pero existen multitud de modelos. Los hay sencillos como el de la imagen o muy complejos, con sistemas de refrigeración auxiliar y adornos varios.

8. **Grifo:** Llave de paso que activa el sistema completo y por el que sale la cerveza. Al igual que la columna, su diseño es muy variado.

9. **Circuito gas:** Conjunto de tuberías por las que circula el gas CO_2.

10. **Circuito cerveza:** Conjunto de tuberías por las que circula la cerveza. Se esteriliza durante la limpieza de la instalación.

Ilustración 6: Esquema instalación de cerveza

Existen en Internet numerosas páginas especializadas que explican detalladamente cada uno de los elementos anteriores y aconsejan sobre su utilización y mantenimiento (8).

2.5 Tecnologías a utilizar

A continuación, se presentan en detalle las distintas herramientas utilizadas para llevar a término el proyecto. Algunas de ellas, cómo los sistemas de gestión Confluence y Jira, forman parte de la organización interna del departamento de proyectos en Athelia. El uso de las mismas contribuye a dar un carácter más profesional al proyecto.

2.5.1 Confluence

Confluence es un software de colaboración para equipos de trabajo. Es propiedad de Atlassian, una empresa dedicada a las aplicaciones para equipos de desarrollo software y administradores de proyectos. Confluence permite centralizar en la web, tareas y documentos relativos a un proyecto. Funciona como un foro común en el que todos los desarrolladores pueden participar. El software desarrollado en Java, se basa en el concepto de wiki. De hecho permite editar las páginas con la conocida notación propia de las páginas wiki.

En el caso de Athelia, se divide el espacio de trabajo en proyectos. Cada proyecto se subdivide a su vez en distintas categorías. En la Ilustración 7, se puede ver un ejemplo de esas subdivisiones. Algunos de los apartados más relevantes son:

- Documentos: Se añaden a este apartado los documentos relevantes asociados al proyecto. En ocasiones se añaden enlaces a los mismos o se transforman a página wiki para ser consultados a través del navegador de Internet.

- Eventos: Se incluyen en este apartado eventos relacionados con el proyecto y que se integran con el calendario de la aplicación, de tal forma que cuando el usuario accede a la plataforma, recibe el aviso de evento.

- Despliegue del software: Para facilitar las tareas de despliegue de una nueva versión, se localizan en esta categoría un histórico de las versiones lanzadas. Esta característica se integra con el software Jenkins tal y como aparece en el apartado 4.5.

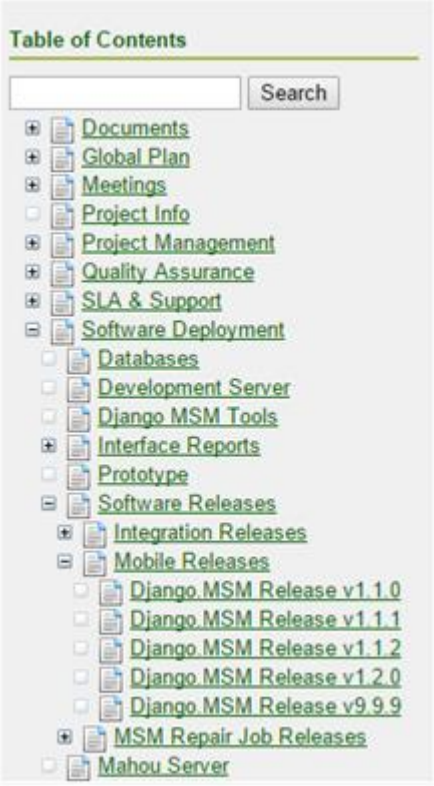


Ilustración 7: Athelia Confluence contents

La herramienta se encuentra alojada en uno de los servidores de Athelia.

2.5.1 JIRA

JIRA es una aplicación web para la gestión de proyectos. Permite realizar planificación de proyectos software y organizar el seguimiento de errores e incidencias. Al igual que Confluence, se trata de una herramienta propiedad de Atlassian. La principal característica de JIRA es la organización de flujos de trabajo.

El software de JIRA permite la integración de procesos con "Subversion". De esta forma, los cambios realizados en un determinado proyecto pueden ser justificados y otros desarrolladores pueden ver los comentarios sobre esas modificaciones.

En el proyecto desarrollado en este trabajo, JIRA ha sido utilizado para organizar y planificar las tareas que debían llevarse a cabo. En el apartado 4.2 se pueden ver más detalles al respecto.

2.5.2 Jenkins

Jenkins es un software que proporciona mecanismos de integración continua durante el desarrollo. La integración continua es un modelo informático que se basa en realizar compilaciones y ejecuciones de pruebas del proyecto de forma automática. Agiliza procesos como la generación de paquetes de instalación y la ejecución de bancos de pruebas.

Jenkins es software libre. Soporta integración con herramientas de control de versiones tipo Subversión. De hecho, los procesos de compilación/prueba normalmente se configuran para ser ejecutados al detectar cambios en los repositorios. Se hablará más sobre Jenkins en el apartado 4.5.

2.5.3 Microsoft Visual Studio 2008 y C# .NET

Microsoft Visual Studio es un entorno de desarrollo para sistemas operativos Windows que permite el desarrollo de aplicaciones basadas en varios lenguajes de programación. Su interfaz y manejo es completamente visual y cuenta con multitud de herramientas de ayuda al desarrollo.

Entre los lenguajes soportados por VS2008 se encuentra C# .NET, que es el lenguaje utilizado para el desarrollo del proyecto. C# es un lenguaje de programación orientado a objetos diseñado por Microsoft. Su sintaxis deriva de C y C++ y forma parte de la plataforma .NET.

Existen una amplia variedad de "plugins" disponibles para VS2008. Para agilizar el desarrollo se ha utilizado un "plugin" llamado ReSharper, desarrollado por JetBRAINS. Esta herramienta permite realizar automáticamente diversas tareas como autogenerar métodos heredados de una interfaz o navegación a través de métodos con un solo clic.

2.5.4 Windows Mobile y Windows Embedded Handheld

Windows Mobile es un sistema operativo para dispositivos móviles desarrollado por Microsoft. El núcleo base del SO es Windows CE (sistema operativo de Microsoft para sistemas embebidos). El sistema Windows Mobile nació como una ramificación de Windows CE y fue lanzando al mercado en el año 2000 bajo el nombre Pocket PC. En el año 2003 cambió de nombre a Windows Mobile.

Llegados a la versión 6.5 de Windows Mobile, Microsoft decidió apostar por un nuevo proyecto de sistema operativo denominado Windows Phone que vio la luz en el año 2010 y que se orientaba más al mercado de consumo (9). Sin embargo, debido a la amplia gama de terminales industriales que incorporaban Windows Mobile y a la escasa aceptación de Windows Phone, Microsoft reconsideró su decisión y lanzó una tercera línea de sistemas operativos para móviles llamada Windows Embedded Handeld (WEEH), que se basaba en el núcleo de Windows Mobile. La tabla siguiente resume la evolución de este sistema operativo:

Tabla 1: Historia sistemas operativos Windows Mobile

Sistema operativo	Núcleo	Fecha de lanzamiento
Pocket PC	Windows CE 3.0	2000
Windows Mobile 2003	Windows CE 4.2	2003
Windows Mobile 6.5	Windows CE 5.2.X	Mayo 2009
Windows Phone 7	Windows CE 6.0	Noviembre 2010
Windows Embedded Handheld 6.5	Windows Mobile 6.5 / Windows CE 5.2.X	Enero 2011
Windows Phone 8.1	Híbrido (NT kernel)	Abril 2014

El sistema operativo Windows Embedded Handeld 6.5 soporta retrocompatibilidad con Windows Mobile 6.5. Esto quiere decir que permite instalar aplicaciones heredadas de Windows Mobile, al contrario que Windows Phone que experimentó una separación total en esa línea.

WEHH domina el mercado de los terminales industriales con un 47% de cuota de mercado (10). La Ilustración 8 recoge una gráfica sobre el mercado de los terminales móviles industriales. Estos datos resultan bastante llamativo si se comparan lo que ocurre en el mundo de los terminales de consumo, donde Android es el ganador indiscutible con un 81.5% de cuota (11).

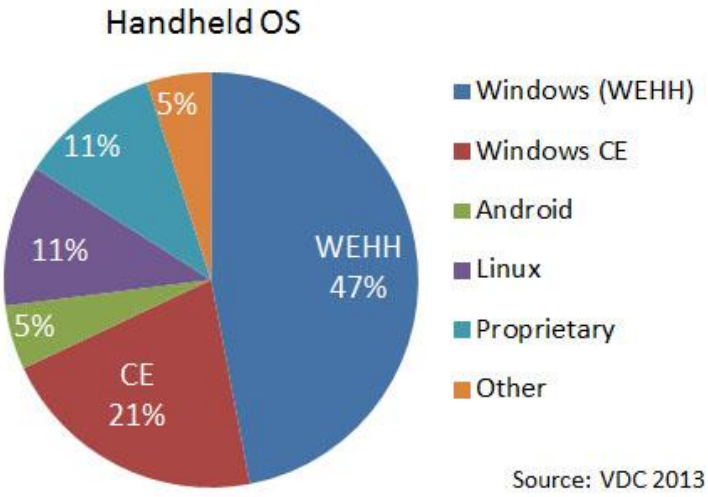


Ilustración 8: Gráfico de mercado Handheld OS 2013

Un aspecto importante de los sistemas operativos integrados en los terminales industriales es que, en general, son sistemas adaptados/modificados por los fabricantes de terminales. Estos fabricantes compran las licencias de Windows CE (núcleo de los sistemas) a Microsoft para luego generar su propia imagen de SO utilizando al herramienta Platform Builder de Microsoft. Por eso, las interfaces y características funcionales pueden variar notablemente de un terminal a otro aun cuando se trata del mismo SO base.

En la Ilustración 9 se puede observar las características de versión del sistema operativo integrado en el terminal objetivo de este proyecto, el Honeywell Black (Véase 2.3.2). Windows Embedded Handeld 6.5 es el sistema operativo incluido en este terminal. El número que aparece acompañando a la etiqueta "Build", se corresponde con la versión generada después de adaptar el núcleo Windows CE 5.2.X.

Ilustración 9: Información build sistema operativo Black

2.5.1 SOTI - Pocket Controller / MobiControl

SOTI es una compañía líder en el desarrollo de aplicaciones MDM (Mobile Device Management). Su software MobiControl es uno de los más importantes del mercado y permite ejercer un control total de los dispositivos incluso de forma remota. Esto incluye seguimiento y localización mediante GPS de cualquier dispositivo que incluya el software y esté vinculado a un determinado administrador.

PocketController es otra de sus plataformas. Este software permite mediante conexión por cable al PC, controlar en tiempo real la ejecución de software en el terminal. Esto supone tener constancia del estado de la memoria o el consumo de CPU en el momento de ejecutar la aplicación que se desee. La Ilustración 10 contiene dos capturas de su interfaz. En el apartado 5 se dan más detalles al respecto.

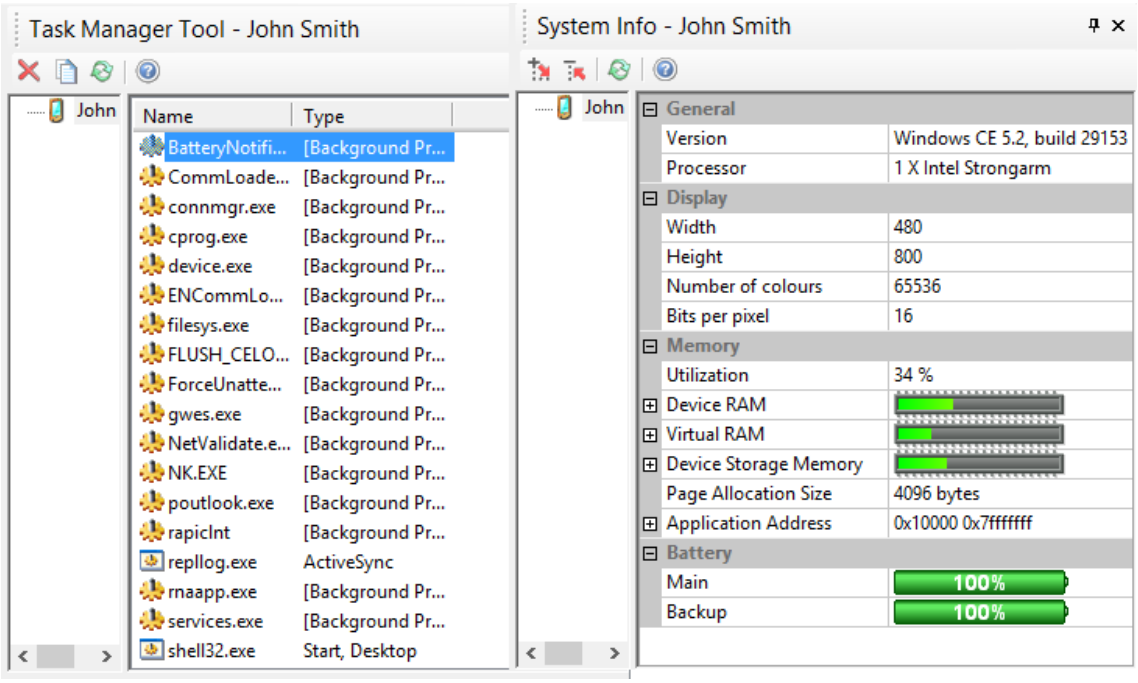


Ilustración 10: Interfaz PocketController - Tasks & Info

2.5.2 MyMobiler

MyMobiler es una aplicación que permite controlar un dispositivo Windows Mobile desde un ordenador con sistema operativo Windows. Esto puede resultar útil para agilizar la copia de archivos entre el PC y el terminal o para utilizar el teclado del ordenador como teclado del dispositivo. Se utilizará esta aplicación principalmente durante la fase de pruebas.

3 Diseño

El diseño es un paso clave en el proceso de desarrollo del software por eso en este apartado se intenta ofrecer una visión lo más completa posible acerca de las características del proyecto.

Los puntos a tratar serán: la gestión del acuerdo, por el cual comenzó el desarrollo software; la arquitectura del entorno de desarrollo integrado Django, explicando en detalle sus características; y por último los diseños detallados en forma de diagramas de flujo del proyecto, en los se pueden apreciar las pantallas de la aplicación.

3.1 Gestión del acuerdo entre Athelia y MSM

Siguiendo la línea expuesta al inicio del documento, Athelia Solutions y Mahou San Miguel comenzaron en 2014 las negociaciones para cerrar el nuevo acuerdo sobre la aplicación de gestión de mantenimiento Django MSM v2.0

El proceso a seguir desde que un proyecto se plantea hasta que comienza su desarrollo es bastante largo y complejo. Athelia Solutions sigue normalmente el mismo esquema de actuación que se detalla en la Ilustración 11.

Como se puede observar en dicho diagrama, el proceso se inicia en el departamento comercial de Athelia. Este departamento prepara una oferta comercial o presupuesto en el que se detallan los trabajos a realizar. Cada tarea se desglosa en subtareas a las que se les asigna una estimación de tiempo en base a unos ratios internos de la empresa. Un ejemplo de este tipo de estimaciones aparece en el Anexo 8.2, donde además se incluye una tabla de tareas desglosadas correspondientes a planificaciones del proyecto.

El siguiente paso corresponde al cliente, que recibe la oferta y la traslada a su departamento de compras. Este departamento debe validar dicha oferta. En muchos casos, el proceso puede detenerse temporalmente debido al estudio que realiza el cliente y que puede incluir modificaciones en la oferta. Por tanto, puede producirse una realimentación del proceso hasta que el cliente esté satisfecho. Esta metodología de desarrollo encaja con el modelo de **desarrollo basado en iteraciones** (metodologías ágiles de desarrollo, Scrum) (12) (13).

Acabado el proceso de gestión del acuerdo, el desarrollo del proyecto está en marcha.
Una vez se produce la validación de la oferta, el pedido entra en Athelia directamente a los responsables de administración de proyectos. Estos generan, manualmente o automáticamente, una orden de fabricación (OF).

La OF es un identificador único del proyecto a realizar y que permite identificarlo a la hora de gestionar las tareas y hacer seguimiento de las mismas. Una vez generada la OF, se integra en la herramienta de gestión empresarial SAGE. A este tipo de software se le conoce con el nombre de ERP, por sus siglas en inglés, *Enterprise Resource Planning*.

La herramienta SAGE ERP X3 permite entre otras cosas, llevar el registro de horas dedicadas al proyecto. Durante el desarrollo del software, se indican día a día el número de horas dedicadas por empleado a los distintos proyectos. De esa forma, al finalizar el mes se tiene registro de todas las actividades realizadas, y es posible analizar la evolución por medio de gráficos estadísticos e informes.

Finalmente, la orden llega al departamento de Proyectos, del que forma parte el autor de este TFG. En este departamento se generan y planifican las tareas en el software de seguimiento de proyectos JIRA. En el apartado 4.2, se ofrece más información sobre esta herramienta.

La mecánica de entrega al cliente sigue otro proceso distinto que se abordará en la sección 5 Pruebas. Resta cumplir los objetivos y ceñirse a la planificación para que el proyecto sea un éxito.

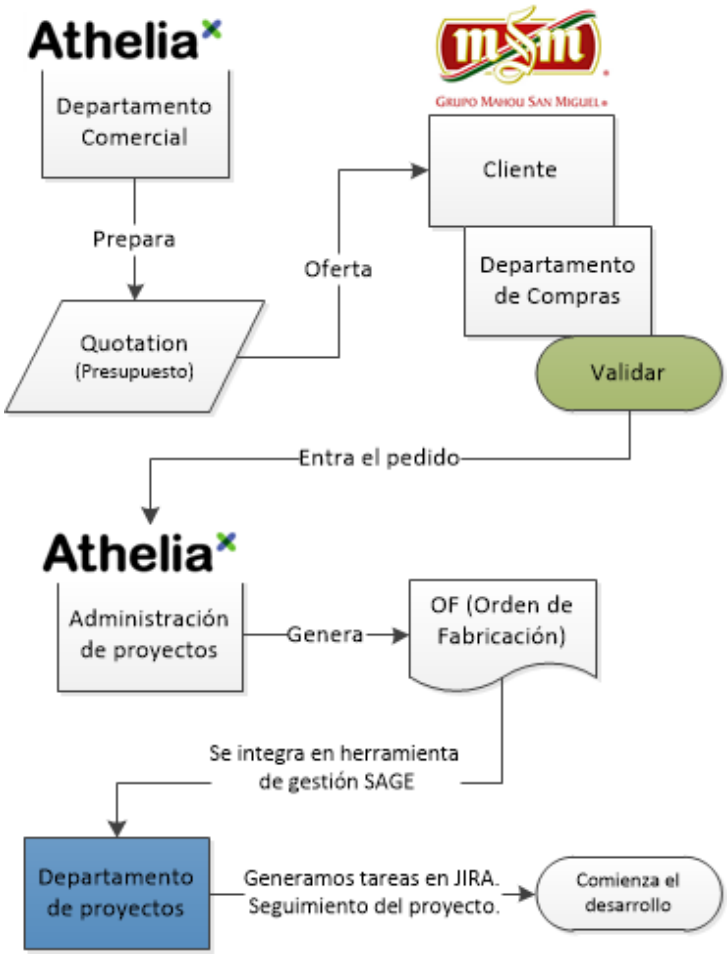


Ilustración 11: Esquema gestión de acuerdo Athelia

3.2 Framework Django

Para entender la aplicación que se detalla en este documento y que constituye el núcleo del TFG presentado, primero hay que comprender que es Django. Durante los meses de trabajo, buena parte del tiempo fue dedicado a comprender el funcionamiento de este software, para poder desarrollar la aplicación final.

El proyecto Django nació paralelo al proyecto para MSM. De hecho, fue la necesidad que planteaba MSM de contar con una aplicación de sólida arquitectura, la que propició el nacimiento de Django, uno de los buques insignia de Athelia.

Las aplicaciones para mantenimiento y trazabilidad de activos guardan muchas similitudes entre sí aunque pertenezcan a clientes distintos. Por ejemplo, las instalaciones de cerveza de MSM son intrínsecamente iguales a las de otras compañías cerveceras como puede ser DAMM (Grupo DAMM). Estas similitudes pueden ser inteligentemente aprovechadas para crear un entorno de desarrollo que integre en sus cimientos esas características.

En conclusión, el objetivo de Django es crear un entorno común para el desarrollo de aplicaciones para terminales móviles industriales, aprovechando las similitudes que presenta el campo del mantenimiento y trazabilidad de activos.

Las ventajas que ofrece contar con un "esqueleto" para el desarrollo de aplicaciones son la causa del éxito de Django. Entre ellas se destacan:

- El desarrollador cuenta con una base de la que partir al elaborar una nueva aplicación, lo que disminuye en gran medida el tiempo de desarrollo, aumenta la eficiencia.

- A menor tiempo de desarrollo, menor es el coste del producto y más proyectos pueden ser abordados al mismo tiempo. Cuantos más proyectos se desarrollan en paralelo, mayor es el número de clientes y aumenta la presencia en el mercado.

- El entorno común facilita la detección y corrección de errores comunes durante el desarrollo del software. El mayor conocimiento del entorno hace que, tanto desarrolladores cómo directivos, asimilen mejor los conceptos de cara a una apertura al cliente.

- Se estandariza el servicio post-venta asociado al desarrollo, y aumenta la capacidad de reacción de cara a posibles evoluciones en el cliente.

Django no es un producto final. No se trata de un paquete software listo para ser usado por el cliente. Lo que Django busca es reducir el esfuerzo de desarrollo mediante la unión de componentes existentes en lugar de reescribir unos nuevos.

3.3 Arquitectura de Django

El modelo de arquitectura del sistema Django es SOA (Service Oriented Architecture). Este paradigma de arquitectura se basa en la idea de construir aplicaciones a partir de la unión de servicios. Esto proporciona flexibilidad y escalabilidad al tiempo que facilita la adición de componentes. A continuación se presenta el diagrama lógico de Django que muestra la organización del sistema:

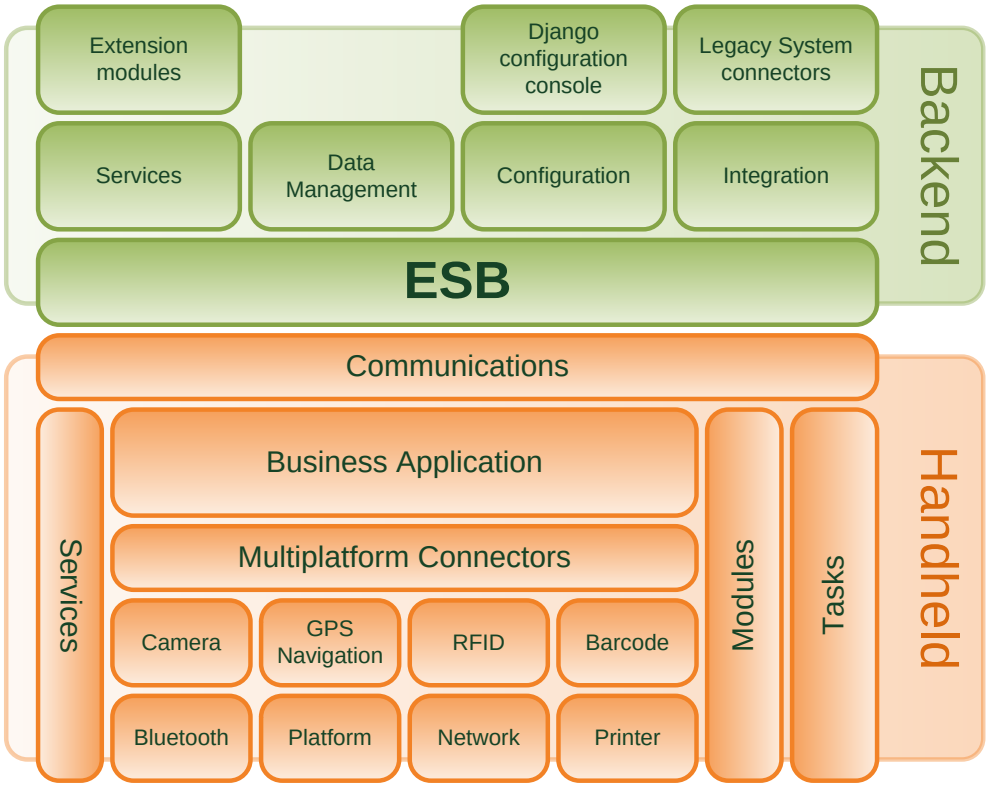


Ilustración 12: Diagrama lógico de Django

En los siguientes apartados se analiza brevemente cada componente:

3.3.1 Backend

El Backend es "el lado servidor". El componente principal que se encuentra en el Backend es el bus de interconexión **ESB** (Enterprise Service Bus). Este componente se encarga de coordinar todas las comunicaciones entre los terminales y el resto de componentes del propio Backend. Estos componentes son:

- Data Management o **Django Central**. Este módulo es el encargado de sincronizar la información entre los terminales y el servidor en ambas direcciones. Recibe datos generados en los handheld, los procesa y los retorna al ESB modificados para que otros componentes como "Integration" los utilicen. También es el encargado de almacenar todos los datos de la aplicación y asegurar la persistencia. La base de datos de Django sigue un modelo definido que no varía de una aplicación a otra.

- **Integration**. Este módulo es el encargado del intercambio de información entre el sistema Django y un sistema externo como un ERP (SAP es un ejemplo de este tipo de software (14)). También realiza la operativa inversa, tomando datos de un sistema externo para incluirlos en Django. Gracias a la utilización del estándar de arquitectura JCA, la conexión con los ERP es más sencilla. El modelo de la base de datos de Django es inmutable, por tanto se debe llevar a cabo una "conversión" de los datos del cliente y viceversa. Esto implica que Django sólo puede ser utilizado en negocios que se adapten a este modelo de datos (15).

- **Configuration**. El módulo de configuración es el encargado de establecer los parámetros técnicos con los que funciona el sistema.

- **Services.** El módulo de servicios permite la adhesión de otros componentes. Estos servicios también utilizarán el ESB con lo que se pueden configurar para comunicarse con cualquier elemento del sistema.

3.3.2 Handheld

El Handheld es "el lado terminal", la interfaz con la que el usuario interactúa. Al igual que en el Backend, se diferencian varios bloques:

- **Business application.** Representa la lógica de negocio particular del cliente que se asienta sobre el resto de componentes. Normalmente se intenta minimizar todo lo posible esta lógica funcional, para utilizar componentes ya incluidos en el framework Django.

- **Services, task and modules**. Los tres elementos hacen referencia a las funcionalidades de la aplicación. Los **módulos** y las **tareas** se relacionan directamente con la aplicación, por ejemplo, mostrar un cierto menú (y su lógica asociada). Los **servicios** son funcionalidades independientes a la aplicación, por ejemplo, la lectura del chip RFID.

- **Multiplatform connectors.** En esta capa se encuentran los elementos que interconectan la lógica de la aplicación con la funcionalidad específica del hardware de los terminales pero sin entrar en las particularidades de cada uno. Un buen ejemplo es la obtención del número de serie del terminal.

- **Communications.** Todas las comunicaciones entre handheld y backend se realizan a través de un broker de conexión y un modelo de publicación/suscripción. La comunicación se realiza enteramente a través de mensajes y eventos lo que permite comunicación en tiempo real con un proceso asíncrono.

La parte "Handheld" recibe el nombre de Django Mobile Client dentro del sistema Django.

3.4 Django Mobile Client

Según el apartado anterior, Django Mobile Client es el componente que se encuentra en el terminal móvil. Debido a que la aplicación desarrollada tiene como base este componente, a continuación se darán más detalles sobre él. Este componente no pretende ser una aplicación en sí misma, sino un entorno de desarrollo que facilita el acceso a herramientas y funciones que se usan comúnmente en el ámbito de las aplicaciones de trazabilidad de activos.

3.4.1 Modelo de aplicación

El modelo de aplicación se basa en los siguientes conceptos que son las entidades básicas con las que trabaja la aplicación y el sistema Django:

- **Activo** ("Asset"): Elemento que se controla de manera independiente. En general, está identificado con una etiqueta RFID.
- **Orden** ("Order"): Tarea/s realizadas sobre un activo en un establecimiento.
- **Ruta** ("Route"): Una ruta de entrega. Es un conjunto de órdenes que siguen una determinada secuencia.
- **Referenciales** ("Reference list"): Se denominan referenciales a los datos asociados a una tarea y que se capturan durante la ejecución de la misma. Por ejemplo, cuando se realiza un montaje de instalación, los materiales que se montan son referenciales.
- **Establecimiento** ("Site"): El lugar físico donde se realiza la Orden.
- **Cliente** ("Customer"): El propietario del establecimiento. Un cliente puede regentar varios establecimientos.
- **Usuario** ("User"): La persona autorizada para utilizar el terminal y por tanto la aplicación. Todo usuario pertenece a un Grupo.
- **Grupo** ("User group"): Conjunto de usuarios de la aplicación. Los grupos tiene asociados un listado de establecimientos.

3.4.2 Orders

Las órdenes se definen como un conjunto de tareas llevadas a cabo sobre un activo en un establecimiento. El contexto de orden ("order-context") es un diccionario de datos manejado en tiempo de ejecución y que comparten todas las tareas. En él se pueden almacenar datos de todo tipo a los que las tareas tendrán acceso. Cuando una orden finaliza se genera una **finished-order,** mensaje en formato xml que vuelca el contenido del order-context y se envía al sistema de Django Central.

Las órdenes se definen utilizando meta información en formato xml en la que se indica el orden de ejecución de las tareas. Este flujo de tareas recibe el nombre de workflow.

3.4.3 Workflows

Los workflows son flujos de ejecución de tareas definidos programáticamente. Su función es realizar un mapeado de conexiones entre tareas. A continuación puede verse un ejemplo de un archivo de este tipo, en el que se definen varios nodos:

```
<node id="change-get-installation-status-node" element-id="change-get-installation-status-task">
  <exit return-value="open" destination="installation-data-node" />
  <exit return-value="close" destination="change-close-installation-alert-node" />
</node>

<node id="change-close-installation-alert-node" element-id="change-close-installation-alert-task">
  <exit return-value="ok" destination="end-workflow" />
</node>

<node id="installation-data-node" element-id="installation-data-task">
  <exit return-value="ok" destination="installation-change-node" />
</node>
```

Como se puede ver en el fragmento de código anterior, los nodos se interconectan entre sí por medio de los valores de retorno, p. ej. `return-value="ok"`, e incluyen un destino, p. ej. `destination="end-workflow"`, que indica hacia donde avanzará el flujo tras la ejecución de la tarea. La traducción de ficheros de workflow y tareas, y el manejo del flujo, es trabajo del núcleo de Django.

3.4.4 Tasks

La tarea es la unidad mínima de trabajo dentro del framework. Se trata sólo de un fragmento de código (una clase "Task") que es manejada con una interfaz estándar que permite ejecutar el código de la tarea y realizar el intercambio de información de entrada/salida de la tarea. Las tareas se definen y configuran según una estructura de metadatos xml con un esquema clave-valor. Para entender mejor su funcionamiento, se expone a continuación un ejemplo de tarea Django.

```
<task id ="set-tag-asset-task" type
="Django.MSM.App.Elements.Tasks.InstallationAssembly.SetTagToAssetTask ,
Django.MSM.App">
  <properties>
    <property key="name" value="Set TAG"/>
    <property key="tagReaderTask" value="assembly-read-tag-task" />
    <property key="movementsDataKey" value="newAssets"/>
  </properties>
</task>
```

La tarea que se define en el fragmento de código anterior, tiene un identificador, una ruta de archivo (que es donde se define el código funcional de la tarea) y una serie de propiedades (son similares a los parámetros de una función).

Todas las unidades de ejecución que se pueden ver en los diseños de la aplicación, son tareas definidas estructuralmente como la tarea ejemplo anterior.

3.5 Acceso a la aplicación Django MSM v2.0

Tal y como se ha explicado en el apartado anterior, Django implementa una estructura sólida para el desarrollo de aplicaciones, pero es sólo la base, por tanto siempre es necesaria una especialización en función de las especificaciones del cliente.

En el caso de Django MSM v2.0, dichas especificaciones fueron definidas en un documento de diseño detallado en el que, a través de numerosas reuniones con el cliente, quedaron cerrados los requisitos funcionales de la aplicación. La mayoría de ellos son los mismos de la aplicación Django MSM v1.0 y sólo algunas operaciones experimentaron ligeras modificaciones. Durante los apartados siguientes, aparecen las funcionalidades principales de la aplicación a desarrollar.

La pantalla de acceso ("login") es un elemento común para una gran cantidad de aplicaciones. En este caso, se trata de un diseño estándar en el que el técnico debe identificarse introduciendo su nombre de usuario y su contraseña (Ilustración 13).

En caso de pulsar sobre el logo de la aplicación, se muestra una pantalla con información de utilidad para identificar el terminal (Ilustración 14)

Una vez realizada la autenticación en el terminal, se muestra un menú principal (Ilustración 15) que permite acceder o bien a la gestión de intervenciones o bien a otras funcionalidades de configuración y administración (ver Funcionalidades generales).

Ilustración 13: Login

Ilustración 14: App Info

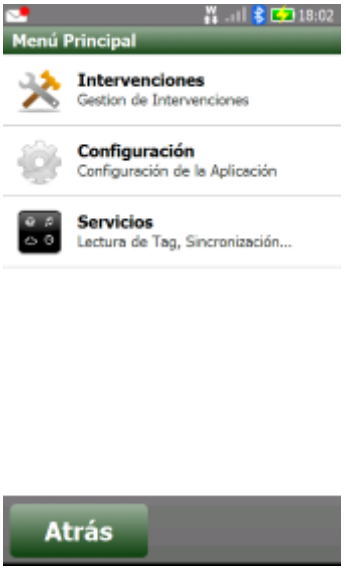


Ilustración 15: Menú principal

3.6 Funcionalidades generales

Las funcionalidades generales, son operaciones estándar que forman parte de la estructura de Django Mobile ya que son comunes a casi todas las aplicaciones que se desarrollan para terminales móviles industriales. Estas operaciones se encuentran en el denominado menú de administración, que se localiza pulsando sobre la barra superior en la pantalla de inicio de la aplicación (pantalla de login); o en el menú de servicios. Las opciones de ambos menús tienen la misma funcionalidad (Ilustración 16).

Ilustración 16: Funcionalidades generales

A continuación se explican con mayor detalle algunas de ellas:

- Configuración de conexión con el servidor: El usuario tiene posibilidad de cambiar dos parámetros:

  - La Dirección IP del Servidor es la IP o nombre del host del servidor de Django Central, necesario para la conexión de la aplicación móvil al sistema Django.

  - El Puerto NMS es el puerto TCP utilizado para la conexión al broker del servidor de Django Central.

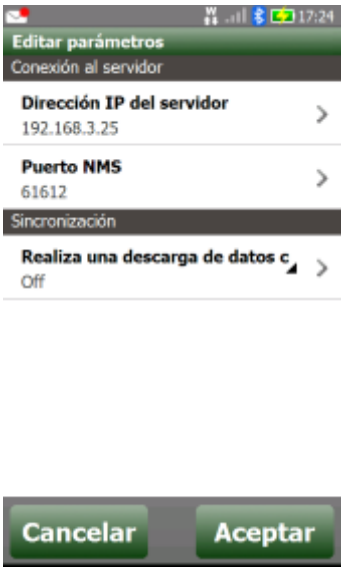


Ilustración 17: Parámetros de configuración

- SOTI: Devuelve un identificador único del terminal para su localización en el software de control SOTI (Opcional según el cliente).

- Electronic id test: Módulo para comprobar el correcto funcionamiento del módulo de lectura de tags.

Ilustración 18: Lectura de tag

- Ping: Verifica la conexión del terminal al servidor central.

- Teclas: esta opción permite la activación o desactivación de ciertas teclas del terminal. Por defecto, las teclas del terminal se encuentran desactivadas para que el usuario no pueda abandonar la aplicación.

- Logger: Mediante esta opción se puede enviar al servidor central el fichero/s de log que el terminal genera a lo largo de su funcionamiento, con el objetivo de permitir al equipo de soporte identificar posibles problemas en el funcionamiento de la aplicación.

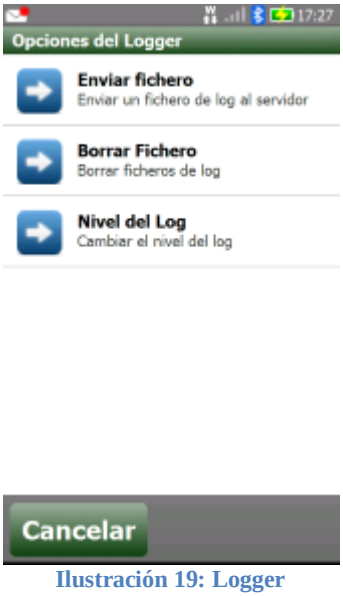


Ilustración 19: Logger

- Actualización de la aplicación: Gestiona el mecanismo para actualizar la aplicación (comprobar nueva versión, descargar, etc).

3.7 Menú de Gestión de Intervenciones.

El menú de Gestión de Intervenciones (Ilustración 20) constituye el menú principal de la aplicación. En él se encuentran las operaciones que el usuario puede realizar sobre las instalaciones de Mahou San Miguel. Se accede a este menú por medio de la opción "Intervenciones" en el menú principal de la aplicación (Ilustración 15), después de realizar la autenticación. Existen tres tipos de intervenciones que se exponen a continuación.

En la versión anterior de la aplicación (Django MSM v1.0), existía un solo tipo de intervención, las Órdenes No Programadas, mientras que en la nueva versión (2.0) van a existir tres tipos, Órdenes Programadas, Órdenes No Programadas y Matriculación. Todas ellas se exponen en los siguientes apartados.

Ilustración 20: Menú Gestión de Intervenciones

3.7.1 Órdenes No Programadas

Las Órdenes No Programadas son aquellas que se realizan sin seguir una planificación preestablecida. En general, este tipo de órdenes se realizan "a petición". Por ejemplo, si un cliente detecta problemas en la instalación presente en su local, puede ponerse en contacto con su técnico para que acuda a solucionarlo.

Este tipo de intervención conserva la funcionalidad de las "Órdenes No Programadas" de la versión anterior de la aplicación (Django MSM v1.0), que se describen en los siguientes apartados.

3.7.1.1 Búsqueda del local:

Una vez seleccionada la opción de Órdenes No Programadas, se lanza un nuevo módulo de búsqueda para que el técnico seleccione el local sobre el que quiere llevar a cabo la intervención. Existen varios criterios para realizar la búsqueda del local.

El módulo de búsqueda también permite añadir filtros múltiples tal como se puede ver en el diagrama siguiente (Ilustración 21). Es un módulo delicado, ya que maneja consultas de cierta complejidad a la base de datos.

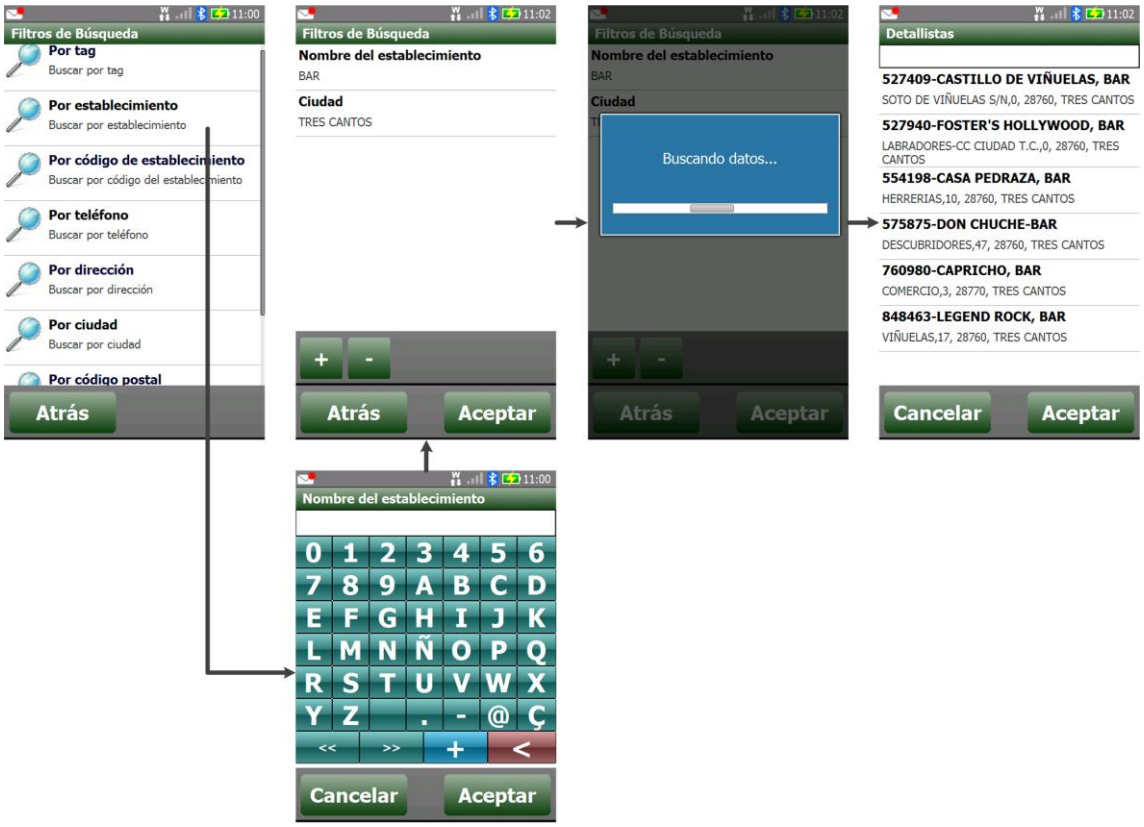


Ilustración 21: Módulo búsqueda de cliente

3.7.1.2 Navegador de instalaciones y Montaje de instalación:

En la última pantalla del módulo de búsqueda, al seleccionar un local, se muestra la pantalla de navegación de instalaciones. Esta interfaz muestra en la parte superior, información abreviada del cliente, y en la parte inferior, las instalaciones presentes en el local del cliente. Se pueden realizar varias acciones en esta pantalla (Ilustración 22):

- Si se pulsa sobre la información del cliente, se accede a una nueva pantalla que muestra información más detallada acerca del cliente y permite modificar algunos datos sobre el cliente (Ilustración 23).

- Si se selecciona una de las instalaciones, se accede al menú de operaciones de la instalación.

Ilustración 22: Navegador de instalaciones

Ilustración 23: Info. Cliente

- Al pulsar el botón "Nueva", se procederá al montaje de una nueva instalación en el establecimiento. No existe un límite preestablecido en el número de instalaciones por establecimiento. Se puede ver el proceso completo de montaje de instalación en el diagrama siguiente:

Ilustración 24: Proceso montaje de instalación

El proceso de Montaje de instalación incluye la selección del tipo de producto junto al cálculo de Ruta (gestionado por la propia aplicación). En el momento del montaje, se deben indicar todos y cada uno de los elementos que la componen. En el apartado 2.4.2 se explicaba en detalle cada uno de estos elementos que se seleccionan a través de listas desplegables en la tercera pantalla del flujo.

3.7.1.3 Operaciones e información de instalación:

Tras seleccionar una instalación de la lista de instalaciones del local se muestra la pantalla de Operaciones (Ilustración 25). En ella se encuentra la lista con los trabajos que pueden ser efectuados. Si se pulsa en la parte superior de la pantalla "Información de la instalación", se accederá a un resumen con los materiales instalados divididos en dos grupos, "Identificados" y "No Identificados" (Ilustración 26). Los elementos "Identificados" son aquellos que tiene asignado un tag identificador.

Ilustración 25: Menú Operaciones no Programadas

Ilustración 26: Info. Instalación

En los siguientes puntos, se exponen las distintas operaciones que pueden llevarse a cabo en la instalación.

3.7.1.3.1 Asignación de un tag a la instalación

En la pantalla que muestra la información de la instalación, existe la posibilidad de realizar una operación denominada "Asignación de tag". Esta operación se utiliza para cambiar el chip RFID asociado al activo de la instalación. Basta con realizar la lectura del nuevo tag para que el cambio tenga efecto.

La operación suele utilizarse en situaciones donde la etiqueta electrónica se ha roto. En caso de que la nueva etiqueta leída se encuentre ya asociada a otro activo (es un caso raro pero puede ocurrir) el sistema mostrará una mensaje de error.

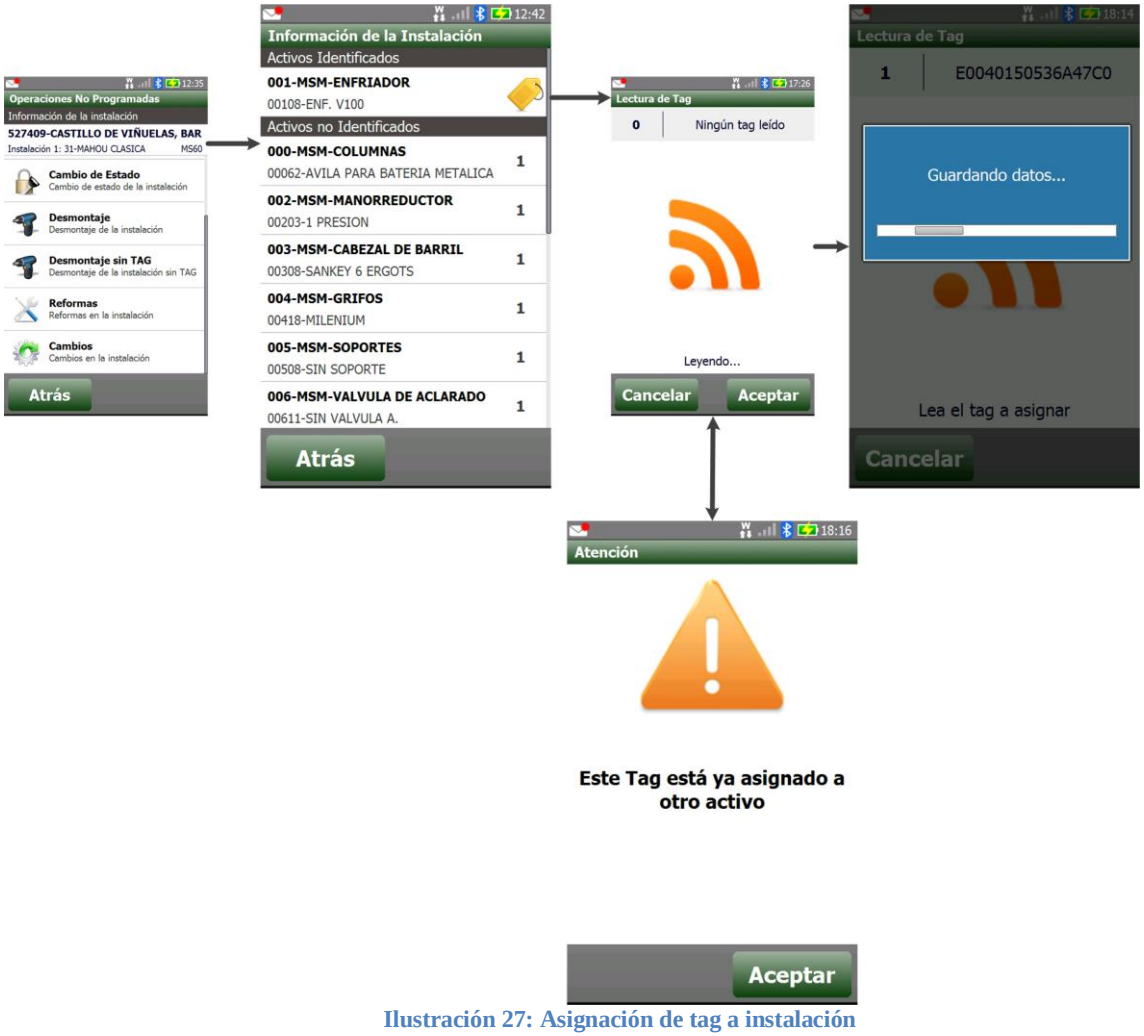


Ilustración 27: Asignación de tag a instalación

3.7.1.3.2 Mantenimiento Preventivo

El Mantenimiento Preventivo es una operación destinada principalmente a realizar labores de limpieza en una instalación, aunque también se asocia a la renovación de líneas del circuito. Las limpiezas suelen ser una actividad periódica y deben llevarse a cabo cada tres meses aproximadamente.

En el momento de realizar esta operación, se indica al usuario si no ha llegado o ha superado el límite temporal establecido para las limpiezas periódicas. Para ello, las instalaciones cuentan con un "contador de limpieza", una variable que indica cuando fue la última vez que se les realizó una limpieza. En el momento de realizar esta operación, el valor de este contador se actualiza.

Para realizar una operación, se añade seleccionándola de una lista con "+". Una vez realizadas todas las operaciones planeadas, se cierran y esto guardará los cambios.

Ilustración 28: Mantenimiento Preventivo

3.7.1.3.3 Mantenimiento Correctivo

Tipo de intervención destinada a resolver las averías detectadas. La resolución de un Mantenimiento Correctivo puede implicar la sustitución de materiales integrantes de la instalación (incluidos elementos etiquetados con tag como el enfriador).

Su mecánica es similar al mantenimiento preventivo, se deben imputar las operaciones realizadas seleccionándolas de una lista que se despliega al pulsar en añadir operación "+". Una vez todas las operaciones han sido imputadas, se procede a cerrarlas.

Si el mantenimiento se realiza sobre un activo (identificado o no), el sistema preguntará si ha sido cambiado. En caso de sustitución, se debe indicar el nuevo material para que quede constancia en la instalación.

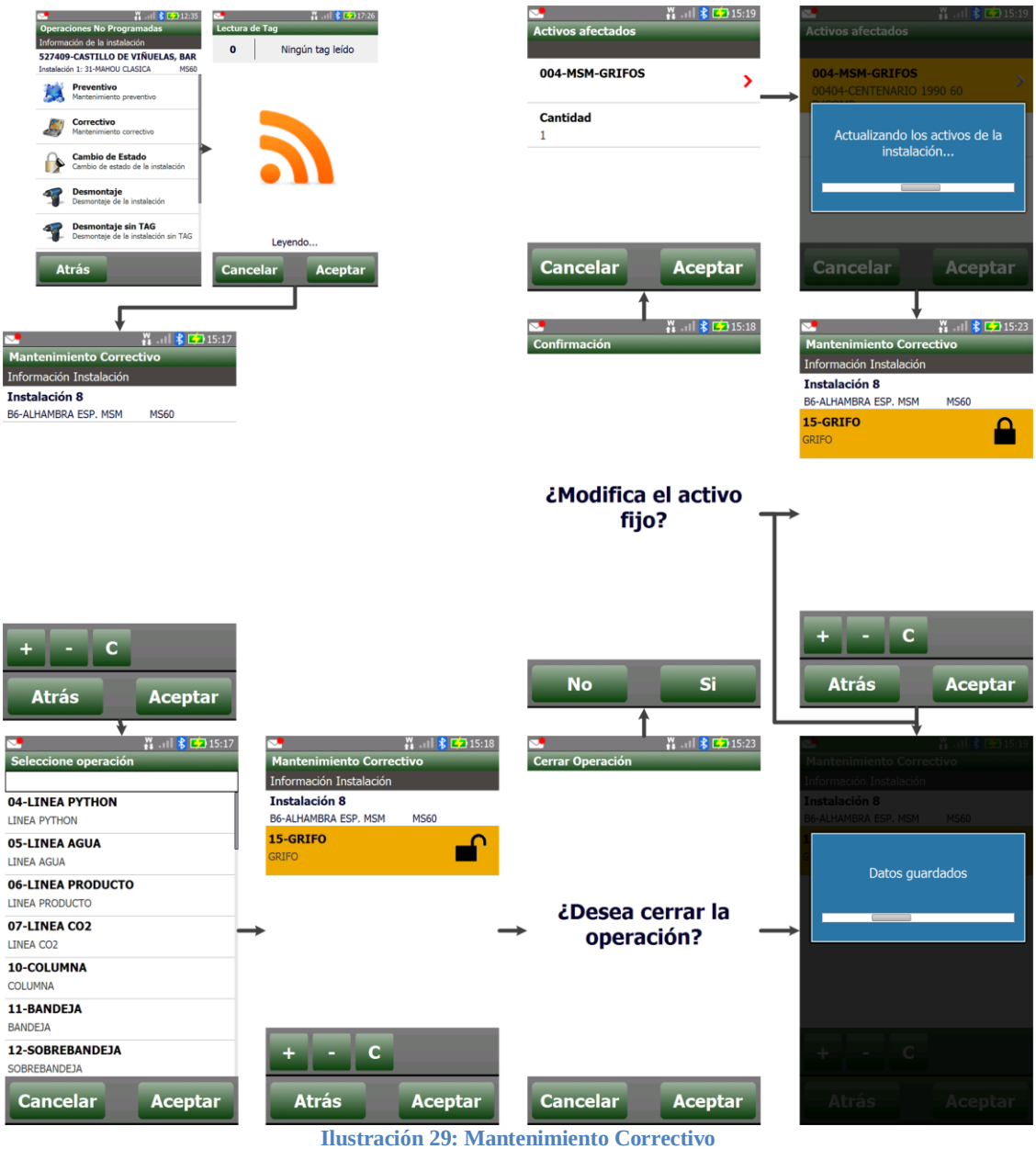


Ilustración 29: Mantenimiento Correctivo

3.7.1.3.4 Cambio de estado de la instalación

Esta operación tiene la finalidad de cambiar el estado de la instalación de "abierta a cerrada" (Ilustración 30) y viceversa (Ilustración 31).

Instalación abierta → cerrada

Ilustración 30: Cambio de estado, Abierta - Cerrada

Instalación cerrada → abierta

Ilustración 31: Cambio de estado, Cerrada - Abierta

El cerrado de instalaciones es común en los locales de carácter estacional, por ejemplo, las terrazas de las piscinas que solo prestan servicio en verano. El desmontaje de dichas instalaciones carece de sentido, ya que el equipo prestará servicio de nuevo a su debido tiempo.

Sobre las instalaciones cerradas no pueden realizarse operaciones. Cuando se intenta realizar alguna operación sobre una instalación cerrada se muestra la pantalla de la Ilustración 32.

Ilustración 32: Error Instalación cerrada

3.7.1.3.5 Desmontaje de la instalación (con/sin tag)

El desmontaje es la operación destinada a retirar todos los elementos de la instalación, que en la mayoría de los casos retornarán a MSM.

Durante su ejecución, se debe indicar el motivo del desmontaje y comprobar que todos los materiales de la instalación retornan en buen estado. Si alguno de los materiales no se encuentra, se restará de los activos retornados y quedará constancia en el sistema.

Existe una problemática con el desmontaje de instalaciones. Puede ocurrir que el técnico reciba la orden de realizar un cierto desmontaje, pero al llegar al establecimiento, encuentre que el activo identificado ha desaparecido o el tag identificador se ha roto. En el caso del desmontaje estándar, no podrá realizar la operación, ya que se le solicita la lectura de dicho tag. Para solucionar esto, se definió una operación de desmontaje que no solicitaba la lectura de tag. El resto de la mecánica de la operación es la misma para ambos casos.

Ilustración 33: Desmontaje de instalación

3.7.1.3.6 Reformas

La operación "Reformas" es aquella en la que el técnico realiza algún tipo de modificación en la instalación. La reforma puede incluir el montaje, desmontaje o sustitución de un material. En caso de que dicha sustitución implique cambiar el tipo de material, se indicará en esta operación.

En el caso de las "Reformas", los materiales sobre los que se realiza la reforma, modifican el stock de materiales, es decir las modificaciones son físicas y quedan reflejadas en el sistema cómo tal. Por ejemplo, si el técnico ha tenido que sustituir la columna del establecimiento por una solicitud del cliente, esa sustitución se facturará como si se comprara el material.

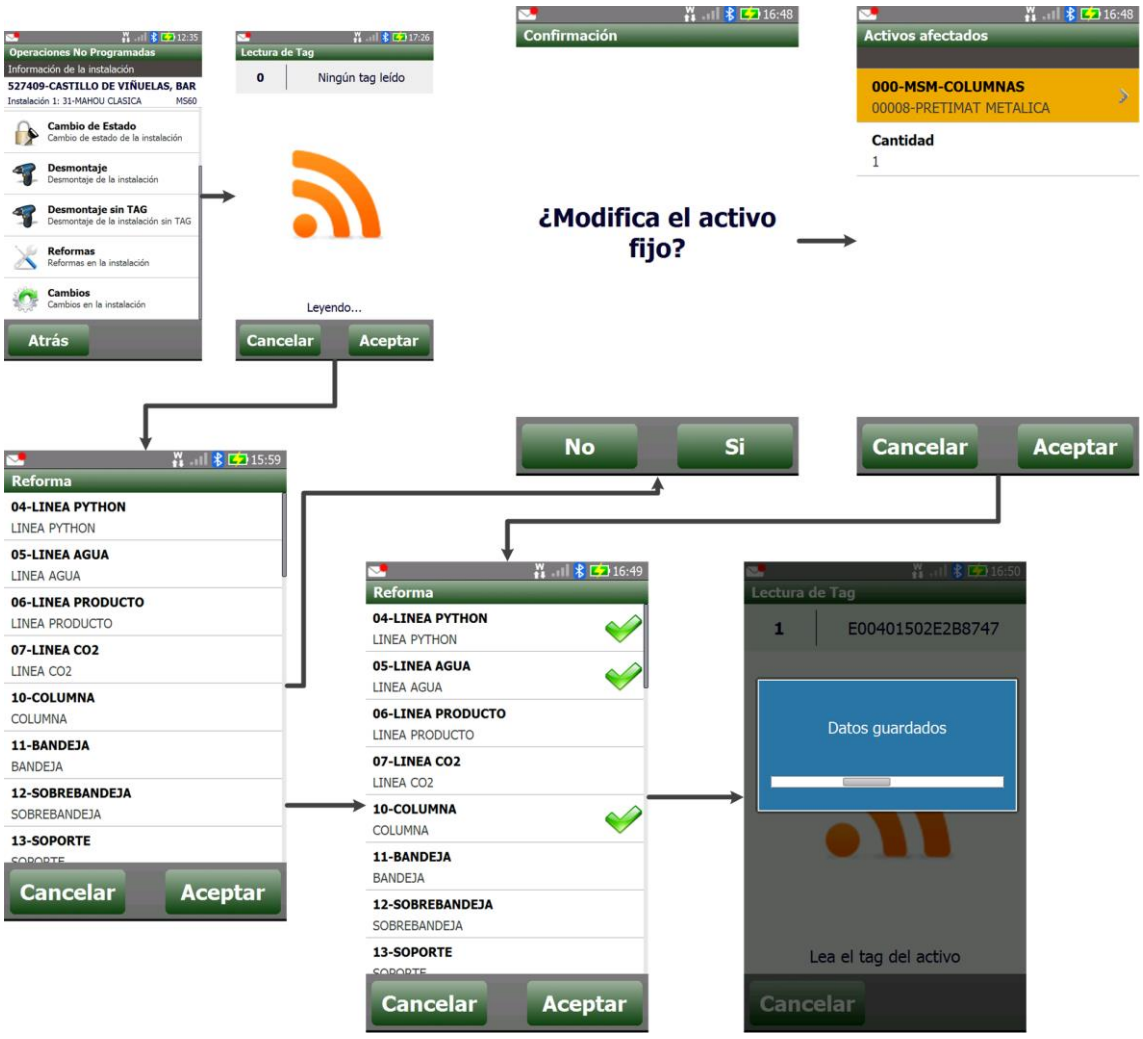


Ilustración 34: Reformas

3.7.1.3.7 Cambios

La operación "Cambios" es parecida a la operación "Reformas" pero tiene una diferencia importante pues la primera es una simple regularización administrativa de los materiales presentes en la instalación. Por ejemplo, si un técnico acude a un bar a realizar una limpieza y encuentra que los materiales reales de la instalación no son los que figuran en el sistema, deberá proceder al cambio de los mismos para mantener la coherencia entre la instalación "real" y la "virtual".

La operación "Cambios", al contrario que la operación de "Reformas", no modifica el stock de materiales, ya que en esencia, solo se trata de cambiar el registro en el sistema, que era incorrecto.

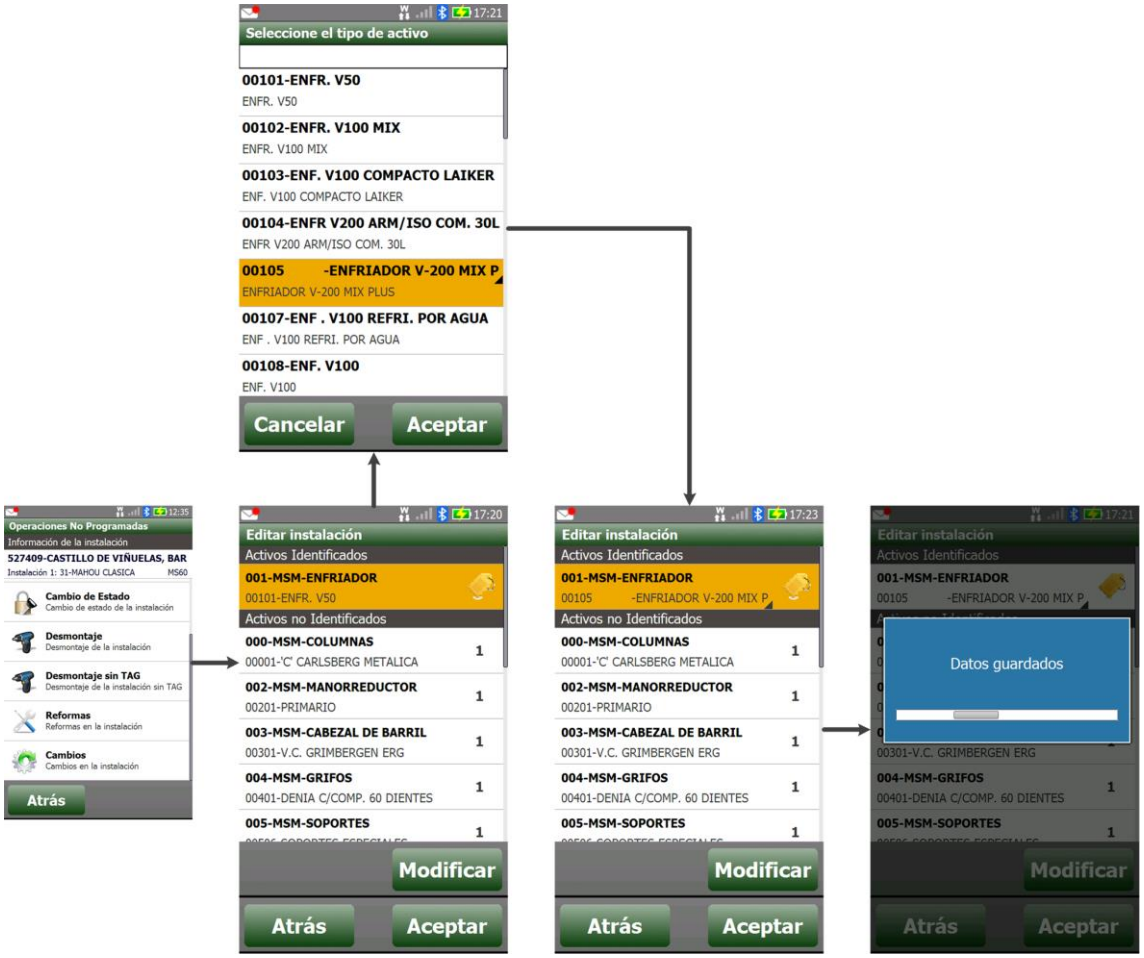


Ilustración 35: Cambios

3.7.2 Matriculación de activos

La matriculación de activos es el proceso por el cuál se identifica un elemento clave de la instalación y se asocia a un tag RFID. Una vez realizada dicha asociación en el ordenador de mano, el tag será considerado a partir de ese momento el nuevo identificador electrónico que permita identificar de forma única e individual el material que lo lleva implantado. Solo algunos materiales son aptos para ser matriculados, en nuestro caso son los enfriadores de cerveza.

En el caso de la matriculación para el proyecto Django MSM v2.0 existe otra funcionalidad adicional, almacenar el número de serie especial propio de los enfriadores de Mahou San Miguel. El diagrama completo del flujo de matriculación puede verse en la Ilustración 36.

Es importante señalar que además de la matriculación "lógica" (a nivel software), el técnico debe llevar a cabo una matriculación "física", es decir, implantar el tag RFID en el material a matricular. Para ello utilizará las herramientas oportunas.

Ilustración 36: Proceso Matriculación de activo

3.7.1 Órdenes Programadas

Las Órdenes Programadas son aquellas que se llevan a cabo siguiendo un plan de trabajo previamente establecido. Este tipo de órdenes no estaba presente en la versión v1.0 y representaba una de las grandes novedades en la aplicación. Permite mejorar la gestión de los grupos de trabajo y los desplazamientos (ver 2.4.1).

Aunque la opción está presente en el menú de Operaciones, no se encuentra operativa ya que las especificaciones y requisitos funcionales de este apartado no están plenamente definidos en el documento de diseño detallado y el desarrollo se pospone para versión futuras del proyecto.

4 Desarrollo e implementación

En los siguientes puntos se tratarán los detalles acerca del desarrollo y puesta en marcha del proyecto. En primer lugar, se trata la cuestión de las modificaciones preliminares, tarea que se llevó a cabo antes de abordar el proyecto y que sirvió de introducción al mismo. En segundo lugar, se expone la planificación inicial con la definición de tareas. A continuación, se explica cómo se creó la estructura de Django Mobile y se detalla la metodología de trabajo con las herramientas de control de versiones Subversion. Después, se ofrecen algunos apuntes sobre mecanismos de integración continua. Finalmente se resumen las diferencias entre la versión Django MSM v1.0 y la versión Django MSM v2.0 para ofrecer una idea exacta de cuál fue el trabajo realizado por el autor del TFG.

4.1 Preliminares: MSM v1.0. Renovación del parque de tags en el cliente.

El estándar de comunicaciones para los sistemas RFID experimentó cambios durante los años que la aplicación Django MSM v1.0 se encontraba en uso. Los tags que MSM tenía instalados en sus equipos, así como lo terminales lectores, seguían las especificaciones de una normativa antigua. Además, dichos terminales habían quedado obsoletos y la empresa fabricante había dejado de producirlos (WAP-3, véase 2.3.1). Todo ello implicaba una renovación total del parque de tags y de terminales móviles por parte de MSM. A partir de este punto, se denominan tags-ARIO a las antiguas etiquetas electrónicas y tags-ISO a las nuevas.

La decisión de MSM fue adquirir un total de 80.000 tags-ISO aproximadamente, estos sí, adaptados a la nueva normativa. Athelia suministró las nuevas etiquetas electrónicas. El plan era que MSM renovara de forma gradual todos sus tags-ARIO antes de empezar a desarrollar la nueva versión de la aplicación móvil (Django MSM v2.0) ya que los nuevos terminales no son capaces de comunicarse con los chips RFID tipo ARIO. Por tanto, la primera tarea era añadir un flujo nuevo a la versión antigua de la aplicación de tal forma que los técnicos pudieran sustituir los tags-ARIO por tags-ISO durante sus visitas rutinarias a las instalaciones.

En este punto se produce la entrada del autor del TFG en el proyecto. La primera tarea asignada fue diseñar el flujo de sustitución de tags. El flujo debía ser intuitivo y no demasiado invasivo, para que el técnico realizara la sustitución de una forma cómoda. Además, MSM solicitó la inclusión de una nueva funcionalidad para la aplicación, la Matriculación[1] de activos. Esta operación tenía la finalidad de realizar la sustitución de los tags directamente en los equipos que no se encontraban instalados en los establecimientos.

[1] La operación Matriculación de activos (3.7.2) no se añadió propiamente en la v 2.0 si no en la v 1.0

4.1.1 Nuevo módulo de lectura de tag

La gran mayoría de las operaciones que ofrecía la aplicación Django MSM v1.0, incluían la lectura del tag identificador de la instalación para facilitar el control de las mismas. Ese módulo de lectura se presentaba como una pantalla que permanecía a la espera de que el técnico aproximara el terminal lector a la etiqueta electrónica.

El nuevo flujo debía incrustarse en mitad de casi todas las operaciones. En la Ilustración 37 se puede ver un ejemplo del flujo de asignación de tag con el nuevo módulo incrustado. Las operaciones afectadas fueron:

- Asignación de Tag a un activo.
- Mantenimiento Preventivo / Correctivo.
- Montaje de instalación
- Reformas.

Ilustración 37: Módulo lectura de tag MSMv1

Cuando el técnico proceda a realizar cualquier tipo de operación sobre la instalación y lea el tag RFID, se analizará su tag. Si el tag es de tipo ARIO, se informará que es necesaria su sustitución por un tag tipo ISO. En ese mismo proceso se mostrará una pantalla para la introducción de la información correspondiente al número de serie del equipo identificado (normalmente el enfriador). En esencia, ese nuevo módulo es muy similar al diagrama de matriculación de activos que aparece en la página siguiente.

4.1.2 Matriculación de activos

Al igual que en el punto anterior, la necesidad de la sustitución de los tags ARIO propició otro cambio importante en la aplicación. Ese cambio fue la inclusión de una nueva funcionalidad denominada “Matriculación de activos”.

La Matriculación de activos es el proceso por el cuál un material/activo sin identificar se vincula con un tag único pasando a formar parte del sistema. Se trata de la misma operación que se exponía en el apartado de diseño 3.7.2.

En la Ilustración 36, se muestra el esquema del módulo de matriculación. Como puede verse, la nueva opción estará disponible a través del menú principal de la aplicación. El esquema es idéntico para el módulo de lectura de tag.

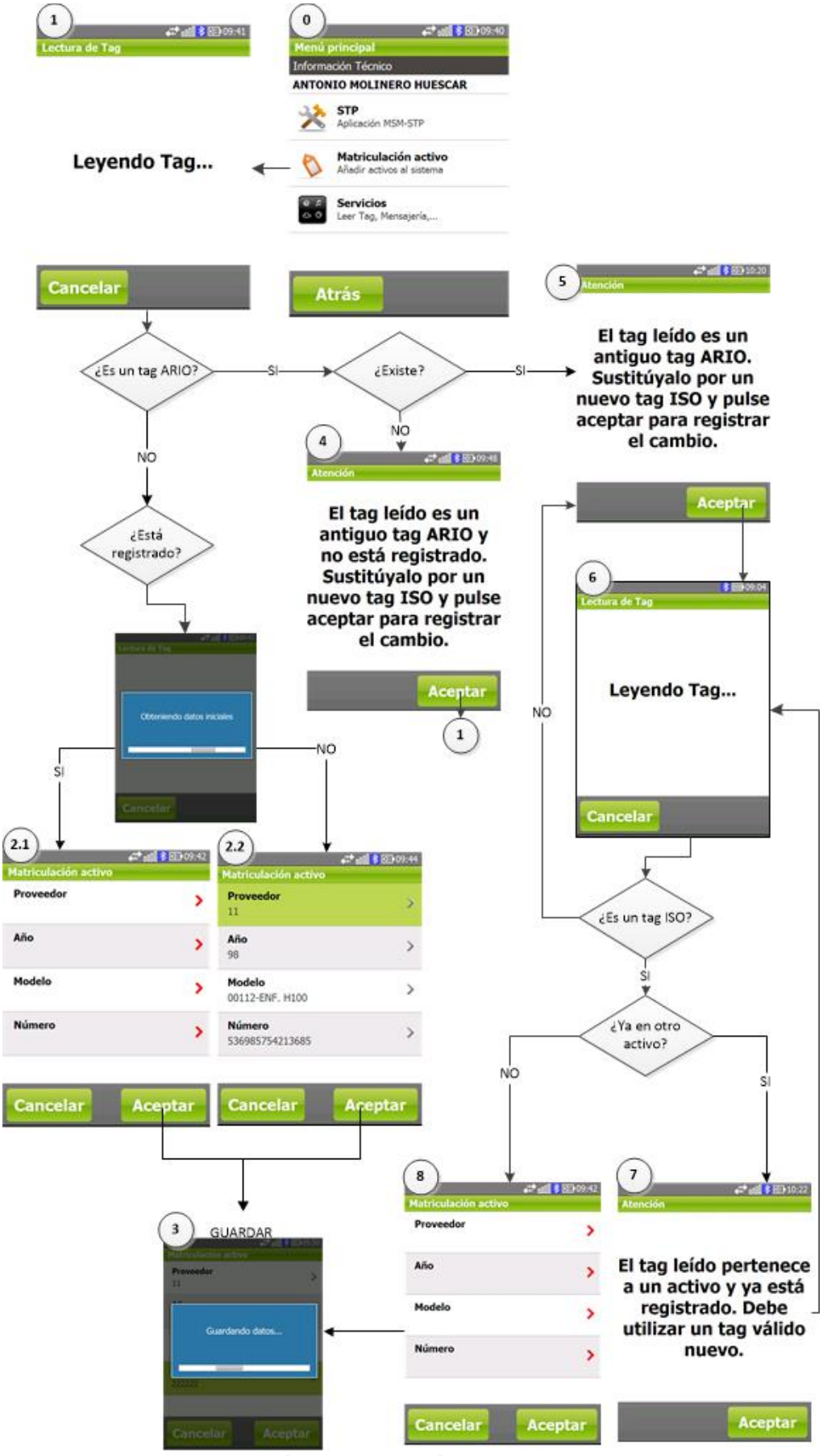


Ilustración 38: Matriculación de activos MSMv1

4.2 Planificación temporal en herramienta JIRA

En el apartado 2.5.1 se presentaba JIRA como una herramienta para planificar proyectos y organizar equipos de trabajo. El primer paso en el desarrollo de un proyecto software es la organización. Los objetivos que se definieron durante el análisis y diseño del software se transforman ahora en tareas concretas a las que se les asigna una duración, un responsable, un revisor ("reviewer"), etc. En el caso del proyecto Django MSM v2.0, la responsabilidad de crear dicha planificación recayó sobre el autor de este TFG.

Siguiendo un flujo de trabajo estándar, un usuario crea una tarea o incidencia dentro de un proyecto existente en JIRA. Esto genera una página dinámica en la que el usuario puede añadir distintos tipos de acciones (tareas, mejoras, bugs). Cada una de ellas está asignada a un usuario de JIRA y tiene un tiempo estimado de desarrollo. Además, se divide el flujo temporal en 4 etapas: "To Do", "In Progress", "To Review", "Done".

En el Anexo 8.2.2 aparece la planificación en JIRA del proyecto en una fecha correspondiente a la mitad del desarrollo. Es interesante la planificación en ese instante ya que se puede apreciar como las tareas "avanzan" a través de las 4 etapas.

4.3 Creación de la estructura de Django Mobile.

Los primeros pasos en el desarrollo de la nueva aplicación incluyen crear la estructura inicial de Django Mobile. El estilo de los proyectos Django es siempre muy similar lo que facilita el desarrollo de nuevos proyectos.

Todo comienza con la creación de una nueva solución en el entorno de desarrollo de Visual Studio 2008. A continuación, se crean los proyectos que la componen y que se explican a continuación:

- **Proyecto principal (Django.MSM):** Este proyecto es el punto de entrada de la aplicación. Contiene la clase "Program" y el fichero de configuración (app.config). La compilación generará un fichero ejecutable de la aplicación "Django.MSM.exe"

    - Program.cs – la clase Program contiene la definición de la función estática Main. En esta función se arranca el denominado DjangoRuntime que representa el framework Django Mobile que a su vez arrancará todos los servicios asociados. Entre ellos se encuentra el servicio de log.

    - app.config – Este fichero XML es de vital importancia. En él se indican variables clave para el funcionamiento del framework Django, entre las que se pueden destacar:

        - resource-assembly – variable que indica la ubicación de los ficheros de recursos.

- loggingConfig – variable que indica el nivel de log de la aplicación (Error, warning, etc) y donde se crea el archivo.

- elements – variable que indica que ficheros xml contienen los esquemas de las tareas utilizadas. Las tareas son un concepto propio de Django que se exponía en el punto 3.4.4.

- workflow – variable que indica el fichero de flujo principal.

- appSettings – apartado donde se indican otras variables de la aplicación, como la dirección IP y el puerto del servidor por defecto, la necesidad de sincronización inicial, etc.

- config-class – variable que indica la clase que contiene las definiciones de configuración para los terminales utilizados. Estas clases son las responsables de la comunicación directa con el hardware en procesos como la obtención del número de serie.

- **Proyecto secundario (Django.MSM.App):** El proyecto secundario tiene el objetivo de evitar problemas relacionados con la gestión de memoria del S.O. del terminal, WEHH (basados en Windows CE .NET). Esta gestión de memoria es bastante particular, en especial el manejo de librerías (DLLs), y los problemas relacionados con la memoria virtual (VM) son bastante frecuentes si no se toman medidas adecuadas durante el desarrollo. Se recomienda la lectura de (16), en donde se explican en detalle la organización, la gestión y los problemas relacionados con la memoria en los sistemas WEHH.

  En este proyecto secundario se sitúan todos los ficheros que componen la funcionalidad de la aplicación y que son:

  - Ficheros “Config”: ficheros XML con las definiciones de los flujos y las tareas que componen la aplicación.

  - Ficheros de teclados y diccionarios: ficheros que contienen los tipos de teclado que utiliza la aplicación y los diccionarios de traducciones para los distintos idiomas.

  - Ficheros “Elements”: Todas las clases implementadas para el proyecto. Estas clases se corresponden con módulos, tareas o funciones de apoyo.

  - Ficheros de recursos: ficheros con las imágenes y sonidos de la aplicación.

- **Proyecto test (Django.MSM.UnitTest):** Proyecto que contiene los módulos de prueba unitarios para la aplicación. Se ofrecen más detalles al respecto en el apartado 5.1.

- **Proyecto servicios (Django.MSM.Services):** El proyecto servicios gestiona los accesos a datos que difieren de los accesos a datos que implementa internamente Django Mobile.

  En Django Mobile existen clases con funciones que manejan el acceso a base de datos, por ejemplo, para añadir una nueva instalación. Estas clases son inmutables, si se quiere modificar su funcionalidad, se debe crear una interfaz que herede de dicha ellas y añadir los métodos que se requieran. Del mismo modo ocurre con los modelos de datos, Django Mobile contempla múltiples modelos de datos predefinidos (por ejemplo, la clase Customer), si se precisa añadir cierta funcionalidad a este tipo de clases, se debe crear una nueva que herede de la clase en cuestión e incluya lo campos o funciones requeridos.

  La idea de gestionar el acceso a datos por medio de las clases de servicio, sigue un patrón de diseño denominado **Facade** (Fachadas), que surge de la necesidad de proporcionar una interfaz simple para un subsistema complejo (17), como sería el de la capa de base de datos. A su vez, la capa de base de datos utiliza otro patrón de diseño denominado **DAO (Data Access Object)**, componente de software que intercala una interfaz común entre la aplicación y el almacenamiento de datos (18). Estos conceptos, que a primera vista parecen sencillos, tienen un enorme potencial ya que permiten al desarrollador expandir la funcionalidad de cualquier método base de Django Mobile y adaptarlo a sus necesidades (19).

- **Proyecto versión (Django.MSM.ProductVersion):** Este proyecto tiene la finalidad de indicar el número de versión de la aplicación que se está desarrollando. Por tanto, debe actualizarse en el momento de generar una nueva versión. Si se añaden correctamente las referencias a este proyecto en el resto de proyectos, se generarán con el número de versión adecuado.

- **Solutions Items -** Django.MSM.build: Este fichero se utiliza para indicar las tareas de compilación al software de integración continua Jenkins. Más información al respecto en el apartado 4.5.

4.4 Control de versiones con Subversion.

Otro de los pasos clave cuando arranca el desarrollo del proyecto, es incluir la solución en la herramienta de control de versiones Subversion. Athelia dispone de un servidor dedicado de tipo Cloud en el que se almacenan los proyectos que se desarrollan.

Dentro del entorno de desarrollo Visual Studio 2008, el usuario tiene a su disposición software tipo plug-in para integrar las herramientas de control de versiones directamente en su proyecto. La aplicación utilizada es Ankhsvn. Su manejo es muy sencillo. Una vez instalada, se integra con Visual Studio para automatizar las operaciones típicas que ofrece una herramienta SVN: añadir solución al repositorio, descargar solución de trabajo (SVN Checkout), control del versionado de archivos, leyenda de archivos modificados, histórico de cambios, etc.

En la Ilustración 39 se puede ver la pantalla que se muestra al seleccionar la solución y escoger la opción "Add Selected Projects to Subversion". Se debe conocer la url destino donde se almacenará el proyecto, y se inluirá en el campo "Repository Url". Automáticamente, VS realizará la conexión con el servidor y mostrará el árbol de ficheros del mismo.

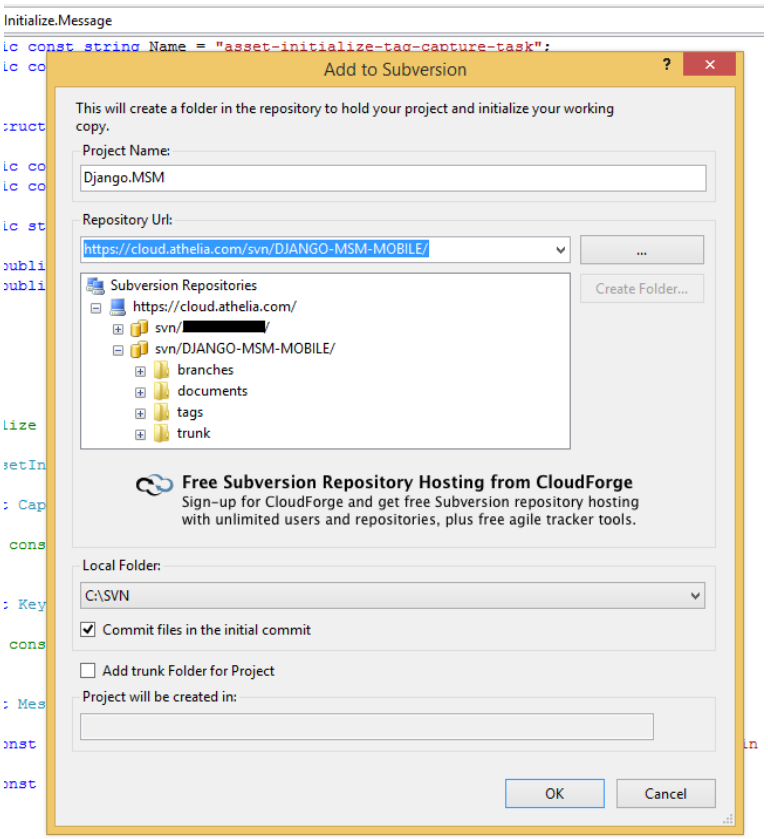


Ilustración 39: Incluir proyecto en Subversion

El manejo de herramientas de control de versiones tiene muchas ventajas. La sincronización que requiere el trabajo en equipo se vuelve mucho más sencilla, ya que una vez se realizan los cambios, el autor puede subirlos al servidor y otro miembro del equipo puede descargarlos con la opción "Update Solution to Latest Version" de VS.

Para subir los cambios realizados al servidor, se utiliza la interfaz de "Pending Changes" que proporciona la herramienta Ankhsvn. En ella se indica que ficheros han sido modificados respecto a la versión que se encuentra en el repositorio. En la Ilustración 40 aparece un ejemplo de actualización de archivos de código fuente.

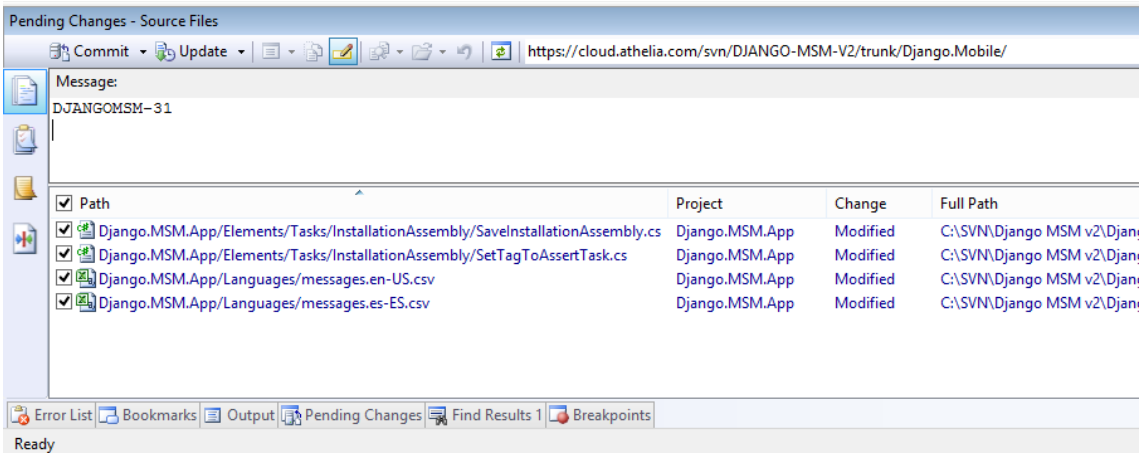


Ilustración 40: Pending Changes

En el campo Message de la interfaz "Pending Changes" se indica un código que corresponde con la tarea generada en JIRA a la que se asocian los cambios realizados. En esta tarea, aparece un resumen de las modificaciones que se realizan a lo largo del tiempo gracias a un proceso de "precommit" configurado en el servidor SVN.

4.5 Software de integración continua Jenkins.

Para conocer mejor las herramientas de control de versiones y su utilidad, se exponen a continuación los pasos que fueron necesarios durante la configuración de Jenkins para este proyecto:

- Se creó, dentro de la solución, un proyecto de tipo Setup donde se indicaban que ficheros serían copiados al terminal durante la instalación de la aplicación. Por cada terminal de despliegue, se debe de generar un proyecto de Setup distinto.

  Dentro de estos proyectos se indicó la ruta de instalación ("Program File\Django.MSM"). Se añadieron a este proyecto los paquetes "Content" y "Primary output" de Django.MSM y Django.MSM.App. De este modo, el sistema automáticamente localiza las referencias del proyecto para copiar las librerías necesarias en la carpeta de instalación.

- Se revisó el proyecto "Product Versión" para asegurar que la versión que se generaría era la correcta.

- Dentro del menú de administración de Jenkins, se definió una nueva tarea:

    - Durante la creación de la misma, se indicó que los datos de configuración serían copiados de una tarea ya existente en el sistema Jenkins. Estos facilitaría la configuración.

    - Se indicó la URL del repositorio donde se aloja el proyecto.

    - Se revisó la versión de NAnt, una herramienta de software libre para configurar procesos de compilación automáticamente (20). Es la versión orientada a .NET de Apache Ant (para Java).

    - Además de la versión, se indicó el nombre del fichero de scripting Django.MSM.build que contiene las definiciones de tareas que realizará la herramienta NAnt para nuestro proyecto. Se puede ver la estructura general de este fichero en el Anexo 8.3 y un ejemplo muy básico de un fichero NAnt en la Ilustración 41: NAnt Hello World.

    - Se configuraron según las necesidades los "Build Triggers". Estos parámetros indican a Jenkins en que momento debe realizar la compilación del proyecto. Existen muchas posibilidades: realizar la compilación periódicamente, sondear el repositorio para que al detectar cambios se lancen las tareas, etc.

    - Los últimos parámetros de configuración añadidos fueron los "Post-build Actions" (acciones que ejecutará Jenkins después de compilar). Estas acciones pueden ser entre otras: enviar un email al administrador con los resultados, generar documentación del proyecto, etc. Otra de las acciones, en este caso característica del proyecto, es generar una página de versión en la herramienta Confluence, apartado Mobile-Realeases.

- Una vez terminada la configuración, se lanzó la compilación manualmente. También es posible esperar a que sea lanzada por los triggers programados.

```
<?xml version="1.0"?>
<project name="Hello World" default="build" basedir=".">
  <target name="build">
    <csc target="exe" output="HelloWorld.exe">
      <sources>
        <includes name="HelloWorld.cs"/>
      </sources>
    </csc>
  </target>
</project>
```

Ilustración 41: NAnt Hello World

4.6 Diferencias entre Django MSM v1.0 y Django MSM v2.0

Para concluir este apartado de desarrollo del proyecto, se van a señalar explícitamente los puntos concretos en los que ha trabajado el autor de este TFG. En definitiva, se expondrán a continuación las diferencias principales que presenta el software Django MSM v2.0 respecto del software base Django MSM v1.0.

4.6.1 Manejo de propiedades dinámicas

Las propiedades dinámicas son un modelo de datos que permite extender cualquier tipo de dato propio del modelo de Django (ver 3.4.1) pero que representa una propiedad particular de una aplicación concreta.

Por ejemplo, un tipo de dato "Cliente" (Customer), puede ser extendido mediante una propiedad dinámica llamada "Teléfono móvil MSM", que indica un atributo especial. Esta propiedad no se incluye en el modelo de datos de Django, ya que no es generalizable a todas las aplicaciones (solo la utiliza la aplicación MSM). Por tanto, ese dato es candidato ideal para formar una propiedad dinámica. Estas propiedades se asocian con su respectivo dato base, mediante un identificador tipo euid-hash, generado aleatoriamente.

Esas propiedades eran gestionadas de forma explícita, en la versión 1.0 de la aplicación. Esto significa que cuando se quería añadir una propiedad dinámica se creaba un nuevo objeto, se insertaba en la base de datos, etc. Esto era así debido a que la versión del núcleo de Django en ese momento solo permitía la gestión explícita.

En el caso de la versión 2.0 esto no funciona de la misma manera, ya que la nueva versión del núcleo de Django implementa otro sistema de manejo de propiedades dinámicas. Este sistema emplea un proceso de carga en segundo plano o **lazy loading** (21), un patrón de diseño que consiste en la inicialización de un objeto en el momento de su utilización y que mejora significativamente el proceso.

Uno de los trabajos realizados fue, repasar toda la aplicación versión 1.0, sustituir el manejo antiguo de propiedades dinámicas por un manejo basado en "lazy loading" y probar que efectivamente el nuevo sistema funciona.

4.6.2 Módulo de lectura de tag incluido en núcleo Django

El módulo de lectura de tag es uno de los pilares fundamentales de esta aplicación. Como se explicó en el apartado de diseño, muchas de las operaciones que se realizan en la aplicación, incluyen la lectura del tag identificador.

Este módulo se configuraba como una tarea independiente en la versión 1.0, situación poco práctica de cara al desarrollo de varias aplicaciones pues tendría que incluirse en todos los proyectos como una clase independiente. Al ser la primera aplicación que utilizaba Django, no se reparó en esta situación.

En la versión 2.0, esta situación cambió ya que el nuevo núcleo de Django incluye entre sus tareas estándar la de lectura de tag. Se expone la transformación en los fragmentos de código siguientes.

```
<task id ="read-tag-task" type
="Django.MSM.Elements.Tasks.ReadTag.ReadTagTask , Django.MSM">
      <properties>
        <property key ="name" value="Read Tag"/>
      </properties>
    </task>
```

```
   <task id="read-tag-task"
 type="Django.Core.Elements.Tasks.ElectronicIdCapture.ElectronicIdCaptureTask,
 Django.Core">
      <properties>
        <property key="name" value="Tag Reading"/>
        <property key="deviceId" value="DDRReader"/>
        <property key="closeOnRead" value="true"/>
        <property key="textConnected" value="Read the tag to assign"/>
      </properties>
    </task>
```

La clave reside en la ruta del fichero que contiene el código fuente que en un caso es interna al proyecto (`Django.MSM.Elements…`) y otra pertenece al núcleo de Django (`Django.Core.Elements…`).

El segundo de los trabajos relevantes realizados por el autor del TFG, fue sustituir la tarea de lectura de tag por la nueva tarea de núcleo, en todos los flujos en los que aparece este módulo,

4.6.3 Módulo de Montaje externo al núcleo de Django

En este caso se da la situación contraria a lo expuesto en el apartado anterior. En la versión 1.0 de la aplicación, el módulo de montaje de instalaciones es interno al núcleo de Django. Esta situación tampoco es lógica ya que el Montaje de instalaciones es una operativa que puede variar de un cliente a otro y no parece muy apropiado situarlo en una capa general.

En la versión utilizada actualmente del núcleo de Django, el Montaje de instalaciones pasa a formar parte de la lógica de negocio propia de cada aplicación. Por tanto, en la aplicación desarrollada en este proyecto (versión 2.0), era necesario incluir este cambio, ya que no se podía acceder a la funcionalidad eliminada del núcleo.

El trabajo en este caso fue progresivo y extenso:

- Primero se analizó el Montaje de instalación presente en la aplicación anterior (versión 1.0). Una vez analizado se recuperó una versión antigua del framework Django, en la que se localizó el montaje de instalaciones.

- Localizada la tarea, se migró a la nueva versión trasladando los archivos, lo que implicó: construir una tarea independiente con sus interfaces asociadas (layouts) y generar un nuevo flujo (workflow). También se debían aplicar los cambios de los puntos anteriores: sustituir el manejo de propiedades dinámicas en la propia tarea recuperada de núcleo, sustituir el uso del módulo de lectura de tag para indicar que ya no pertenecía al núcleo, realizar los cambios en los tipos de datos utilizados adaptándose al nuevo modelo, etc.

- Por último, se llevaron a cabo otras modificaciones variadas como la sustitución de cadenas fijas por constantes y diversas prácticas para mejorar legibilidad del código.

El hecho de que ciertas funcionalidades se modificaran/desaparecieran de una versión a otra del núcleo de Django se repitió con bastante frecuencia. Lo que implicaba repetir los pasos anteriores según el caso.

4.6.4 Cambios en el modelo de datos de Django

Uno de los puntos más críticos de la actualización de la aplicación son los cambios en el modelo de datos interno de Django. En el apartado 3.4.1 Modelo de aplicación, se detallaban las principales entidades con las que trabaja el sistema Django. Además de las expuestas en ese punto, existen algunas otras como el tipo de activo (AssetType), el tipo de instalación (InstallationType), etc.

En el caso del AssetType e InstallationType, la versión 1.0 del software las manejaba como las entidades que representan. Esto quiere decir que contaban con su propia clase de servicios que gestionaba el núcleo de Django (A modo de recordatorio, las clases de servicio son clases que actúan como fachadas para el acceso a datos). Sin embargo, en la nueva versión de núcleo estas clases de servicio desaparecieron. Esto se debe a que se decidió que, en lugar de almacenar los objetos en memoria, era más eficiente almacenar solo referencias a ellos (Ids).

Por ejemplo el AssetType propio del Asset, se almacenaba dentro del objeto Asset. En la nueva versión, sólo se almacena un número (Id) de referencia. Si el desarrollador decide acceder a las características del Assettype de un determinado Asset, se realiza un proceso de lazy-loading utilizando el id almacenado en el objeto.

Otro de los trabajos significativos realizados por el autor de este TFG consistió en recorrer toda la aplicación versión 1.0 y modificar el acceso a las entidades de servicios que habían desaparecido. El cambio incluía añadir la nueva gestión. Por ejemplo, en el caso del activo, el campo AssetType ya no era un objeto si no un Id y era necesario indicarlo.

5 Pruebas

En este apartado se describen las distintas pruebas que se llevaron a cabo para certificar la calidad del software desarrollado. En este caso, la ejecución de pruebas no se pospuso hasta el final del proyecto, si no que durante el desarrollo, conforme se añadían funcionalidades a la aplicación, se probaba modularmente que su funcionamiento era correcto. A pesar de ello, es importante realizar una batería final de pruebas documentadas para no dejar ningún cabo suelto.

5.1 Test unitarios

Los test unitarios son pruebas realizadas sobre el funcionamiento de un módulo de código. Su finalidad es asegurar el funcionamiento de los módulos por separado. Normalmente, estas pruebas se enfocan al testeo de funciones o clases.

Para llevar a cabo este proceso de pruebas, se utiliza un framework para lenguajes basados en .NET denominado **NUnit** (22). Esta herramienta open-source permite la ejecución de pruebas unitarias directamente en el entorno de desarrollo. Se basa en el concepto de "Assertions" o comprobaciones de valor (AreEqual, AreNotEqual, IsTrue, IsFalse, etc.). Cumple las mismas funciones que el conocido JUnit (23) para el mundo de Java y ambos pertenecen a la familia xUnit.

Tal y como aparecía en el apartado 4.3, en la estructura de Django Mobile se incluye un proyecto específico para la realización de test unitarios. En este proyecto se localizan las clases de test y los "mocks". En programación orientada a objetos, se denomina "mock" a un objeto simulado que imita el comportamiento de un objeto real pero construido según las necesidades del tester. Se utiliza para simular comportamientos en pruebas unitarias (24).

En el Anexo 8.1 se puede ver un ejemplo de test unitario desarrollado para el proyecto. Cuando finaliza la generación de test unitarios, se procede a su ejecución. El entorno VS2008 integra NUnit y permite ejecutar los test mediante la creación de "sesiones de test" (Ilustración 42)

La mayoría de test que se integran en el proyecto, iban destinados a probar clases desarrolladas en el proyecto de "Services" (ver apartado 4.3). La razón es que las clases de servicio son en general módulos independientes que pueden utilizarse en otros proyectos, lo que requiere comprobaciones exhaustivas. El testeo unitario de otras funciones, como las clases de "tasks", es más complejo ya que presentan una interdependencia fuerte con otros módulos. Por tanto, se comprobó su funcionamiento mediante pruebas de integración al desplegar la aplicación en el terminal objetivo.

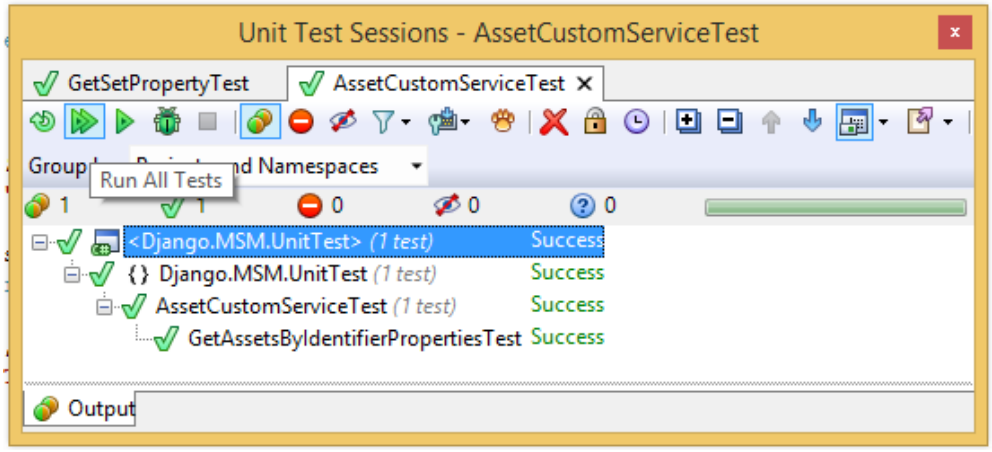


Ilustración 42: Sesiones de test VS

5.2 Despliegue e instalación en el terminal

El despliegue e instalación en el terminal es una operativa que se sigue durante el desarrollo del software. A efectos prácticos, tiene la misma finalidad que los llamados test de integración, que se utilizan para probar el funcionamiento del conjunto de módulos de la aplicación.

El funcionamiento de los módulos por separado no implica el funcionamiento conjunto, de ahí el interés de este tipo de pruebas. En nuestro caso, se puede ver un resumen de dichos test en la Tabla 2.

Una vez compilada la aplicación, se realiza el proceso de despliegue. Para controlar más fácilmente el terminal, se utiliza la herramienta MyMobiler (veáse 2.5.2), y para monitorizar los procesos y comprobar el estado de la memoria, se utiliza PocketPC (veáse 2.5.1). VS2008 permite desplegar ("Deploy") la aplicación en un terminal conectado al pc por medio del "Windows Mobile Device Center". Este programa, además de instalar los drivers correspondientes para comunicarse con el dispositivo, establece una conexión segura con el mismo.

Las labores de prueba se agilizan significativamente gracias al servicio de log de la aplicación que registrar cualquier posible error, volcando la información en un fichero de texto. Así, en el momento en que se produce un comportamiento inesperado, se puede acudir a dicho fichero para ver que ha ocurrido.

Test	Resultado	Corrección – Comentarios
Menú de administración		
Opción "Configuración"	Correcto	Cambio de IP, puerto y sinc. Inicial correcto.
Opción "SOTI"	Correcto	La opción no muestra número de identificación hasta que se defina si es finalmente incluida.
Opción "Electronic Id"	Correcto	Módulo de lectura de Tag funcional, utilizado en el resto de operaciones.
Opción "Ping"	Correcto	Funcionalidad correcta.
Opción "Teclas"	Correcto	Funcionalidad correcta.
Opción "Logger"	Correcto	Cambio de nivel de Log correcto
Opción "Actualización"	Correcto	Proceso de actualización no definido.
Pantalla de login		Muestra logo correctamente
Acceso a la aplicación	Correcto	Conexión con el servidor correcta. Descarga inicial de datos configurada.
Error usuario	Correcto	Muestra el error.
Error contraseña	Correcto	Muestra el error.
Acerca de - SN	Correcto	Muestra correctamente el número de serie del terminal.
Menu principal	Correcto	Las opciones se muestran correctamente.
Intervenciones	Correcto	Las opciones se muestran correctamente.
Órdenes programadas	Correcto	La opción no es plenamente funcional en esta versión de la aplicación.
Órdenes no programadas		
Filtros de búsqueda		
Por establecimiento	Correcto	Búsqueda correcta.
Por código	Correcto	Búsqueda correcta.
Por teléfono	Correcto	Búsqueda correcta.
Por dirección	Correcto	Búsqueda correcta.
Por ciudad	Correcto	Búsqueda correcta.
Por código postal	Correcto	Búsqueda correcta.
Por propietario	Correcto	Búsqueda correcta.
Navegador de instalaciones	Correcto	Muestra correctamente información del cliente (falta un campo), instalaciones disponibles y estado de instalación.
Info. cliente	Error	No se permitía el acceso a la información debido a un problema de configuración de módulo en Django. Corregido en Core.
Modificación info. cliente	Correcto	Modificaciones persisten en DB

Nueva instalación		
Cálculo de ruta	Correcto	Se calcula la ruta del técnico automáticamente.
Montaje de componente	Correcto	Se seleccionan los componentes de listas desplegables generadas mediante consulta SQL.
Guardado de datos: Activo, Activos asociados e instalación	Error	El activo no era guardado en la base de datos. Se añadió proceso de guardado de activo.
Operaciones		
Preventivo		
Añadir operación	Correcto	Se añade correctamente la operación.
Comprobación fecha limpieza	Correcto	La fecha de limpieza no se muestra actualizada en el navegador de instalaciones.
Guardar datos	Correcto	Datos comprobados en DB.
Correctivo		
Añadir corrección material	Correcto	Se añade correctamente la operación.
Cierre / Modificación material	Correcto	Se solicita nuevo activo y persiste el cambio.
Guardar datos	Correcto	Datos comprobados en DB.
Cambio de estado		
Abierta – Cerrada	Correcto	Detecta abierta, cambia a cerrada.
Nav. Inst. muestra cierre	Correcto	Se muestra imagen de abierto-cerrado.
Bloqueo de operaciones	Correcto	No se permite realizar operaciones en instalaciones cerradas.
Cerrada – Abierta	Correcto	Detecta cerrada, cambia a abierta.
Desmontaje / sin TAG		
Modificación materiales	Correcto	Permite seleccionar si se recuperan materiales.
Guardar datos	Correcto	Datos comprobados en DB.
Reformas		
Añadir reforma material	Correcto	Se añade correctamente la operación.
Guardar datos	Correcto	Datos comprobados en DB.
Cambios		
Edición de instalación	Correcto	Permite edición de cualquier material.
Guardar datos	Correcto	Datos comprobados en DB.
Matriculación		
Formulario número de serie	Correcto	Solicita campos nuevos correctamente.
Guardar datos	Correcto	Datos comprobados en DB.

Tabla 2: Test unitarios

Una vez finalizadas las pruebas en el terminal, la aplicación está lista para ser empaquetada y entregada al cliente.

6 Conclusiones

El negocio de la trazabilidad de activos se consolida en el ámbito industrial y al mismo tiempo, el mercado de los terminales móviles industriales avanza rápidamente, como casi todo lo relacionado con tecnología. La adaptación a estos cambios es vital para asegurar el crecimiento de una compañía e implica en muchos casos una renovación sustancial. Mahou San Miguel lo sabe, y llevó a cabo esta renovación en dos ámbitos diferentes, por un lado, la renovación de los tags asociados a activos instalados en los establecimientos de clientes, y por otro lado, la actualización de la aplicación que gestiona las operaciones de mantenimiento llevadas a cabo por técnicos en instalaciones de venta de cerveza.

Athelia Solutions fue la empresa encargada del desarrollo de esta nueva aplicación que recibió el nombre de Django MSM v2.0. La aplicación se instaló en un modelo de terminal más moderno que el utilizado en la anterior versión.

El trabajo llevado a cabo y que se ha expuesto a lo largo del documento, tenía como finalidad diseñar, desarrollar y probar la nueva aplicación. El diseño, basado en el framework Django, conservó las funcionalidades principales de la versión 1.0 de la aplicación y añadió otras nuevas. El desarrollo, se llevó a cabo siguiendo una metodología ágil por iteraciones y con la ayuda de herramientas de desarrollo como JIRA, Subversion y Jenkins. Las pruebas aseguraron la calidad del software según los módulos desarrollados y su funcionamiento conjunto, lo que implicaba la instalación en el terminal de despliegue.

El fruto del trabajo ha sido una aplicación plenamente funcional, adaptada a las necesidades del cliente y preparada para ser desplegada en un entorno real.

6.1 Trabajo futuro

Como trabajo futuro se propone:

- La inclusión de las Operaciones Programadas. Aunque la funcionalidad última de las operaciones será exactamente la misma que la de las operaciones no programas ya desarrolladas, será necesario añadir la lógica de programación de tareas (calendarios, búsqueda de órdenes por día, etc.).

- La migración de la aplicación a sistemas Android. Este cambio se podría llevar a cabo a través de la utilización de software Xamarin (25), que permite el desarrollo de aplicaciones multiplataforma en C# (compatible con .NET).

- La adaptación de la aplicación a otros modelos de terminales además del terminal objetivo actual. Sólo se precisa modificar algunas configuraciones.

7 Referencias

1. **Athelia.** Athelia.com. *Athelia.* [En línea] 2015. [Citado el: 20 de Abril de 2015.] http://athelia.com/comunidad/atheliasolutions.

2. **Bonsor, Kevin y Fenlon, Wesley.** How RFID Works. *HowStuffWorks.* [En línea] InfoSpace LLC, 05 de November de 2007. [Citado el: 03 de Mayo de 2015.] http://electronics.howstuffworks.com/gadgets/high-tech-gadgets/rfid.htm.

3. **Knuth, Peter y Modrák, Vladimír.** Traceability of Material Flow by use of RFID Technology. [En línea] 2012. [Citado el: 10 de Abril de 2015.] http://www.fvt.tuke.sk/journal/pdf07/2-str-92-95.pdf.

4. **Psionet.** Psion.com. *PSION.* [En línea] 01 de Enero de 2010. [Citado el: 01 de Mayo de 2015.] http://www.psion.co.za/workabout_pro_handheld_computer_terminal.html.

5. **Honeywell International Inc.** Honeywell | Scanning & Mobility. *Honeywell.* [En línea] 01 de 01 de 2015. [Citado el: 28 de Abril de 2015.] http://www.honeywellaidc.com/en-US/Pages/Product.aspx?category=hand-held-mobile-computer&cat=HSM&pid=dolphin70e.

6. **Mahou San Miguel.** Quiénes somos. *Mahou SanMiguel.* [En línea] 2014. [Citado el: 02 de 05 de 2015.] http://www.mahou-sanmiguel.com/es-es/sobre-nosotros/quienes-somos.html.

7. **KegWorks.** How draft systems work. *KEGWORKS.* [En línea] 21 de Agosto de 2012. [Citado el: 28 de Abril de 2015.] http://www.kegworks.com/blog/how-draft-systems-work/.

8. **Técnica Cervecera.** Blog de la cerveza. *Técnica Cervecera.* [En línea] 2013. [Citado el: 06 de Abril de 2015.] http://tecnicacervecera.blogspot.com.es/.

9. **notebooks-AMY.** A brief history of Windows Mobile. *notebooks.com.* [En línea] 2010. [Citado el: 11 de Mayo de 2015.] http://notebooks.com/2010/04/12/a-brief-history-of-windows-mobile/.

10. **El Mundo del ADC.** El mejor SO para movilidad. *El Mundo del ADC.* [En línea] 20 de Enero de 2015. [Citado el: 08 de Mayo de 2015.] http://www.elmundodeladc.com/el-mejor-so-para-movilidad/.

11. **FM, Yúbal.** Android consolida su liderazgo en 2014. *Xataka Android.* [En línea] 25 de Febrero de 2015. [Citado el: 06 de Mayo de 2015.] http://www.xatakandroid.com/sistema-operativo/android-consolido-su-liderazgo-en-2014-alcanzando-el-81-5-de-cuota-de-mercado.

12. **Proyectos Ágiles.** Desarrollo iterativo e incremental. *proyectosagiles.com.* [En línea] [Citado el: 25 de Abril de 2015.] http://www.proyectosagiles.org/desarrollo-iterativo-incremental.

13. —. Qué es SCRUM. *proyectosagiles.com.* [En línea] [Citado el: 25 de Abril de 2015.] http://www.proyectosagiles.org/que-es-scrum.

14. **SAP.** About SAP. *SAP.com.* [En línea] SAP, 2015. [Citado el: 28 de Abril de 2015.] http://go.sap.com/about.html.

15. **Magro, Conversión, González, David y Rodríguez, Pablo.** *Django General Design.* Madrid : Athelia, 2010.

16. **Robinson, Reed.** Slaying the Virtual Memory Monster. *MSDN Blogs.* [En línea] 2007 de Agosto de 31. [Citado el: 12 de Mayo de 2015.] http://blogs.msdn.com/b/hegenderfer/archive/2007/08/31/slaying-the-virtual-memory-monster.aspx.

17. **Gamma, Erich, y otros, y otros.** Structural Patterns - Facade. [aut. libro] Erich Gamma. *Design Patterns: Elements of Reusable Object-Oriented Software.* s.l. : Prentice Hall, 2002.

18. **Alverik.** Patrón DAO (Data Access Object). *rinconprogramadorcito.blog.* [En línea] 05 de Noviembre de 2010. [Citado el: 10 de Mayo de 2015.] http://rinconprogramadorcito.blogspot.com.es/2010/11/patron-dao-data-access-object.html.

19. **Azaustre, Carlos.** Cómo utilizar el patrón Facade (Fachada) en bases de datos. *Software y otras desvirtudes.* [En línea] 20 de Septiembre de 2012. [Citado el: 10 de Mayo de 2015.] http://softwareyotrasdesvirtudes.com/2012/09/20/como-utilizar-el-patron-facade-fachada-en-bases-de-datos/.

20. **McManus, Jeffrey.** Using NAnt to Build .NET Projects. *windowsdevcenter.com.* [En línea] 27 de Enero de 2003. [Citado el: 08 de Mayo de 2015.] http://www.windowsdevcenter.com/pub/a/dotnet/2003/01/27/nant.html.

21. **Miguel, César Pérez.** El patrón de diseño Lazy Loading. *adeshoras.com.* [En línea] 2009. [Citado el: 13 de Mayo de 2015.] https://adeshoras.wordpress.com/2008/04/20/el-patron-de-diseno-lazy-loading/.

22. **Nunit.** What is NUnit? *NUnit.* [En línea] 2015. [Citado el: 03 de Mayo de 2015.] http://www.nunit.org/.

23. **junit.** About JUnit. *JUnit.* [En línea] 2014. [Citado el: 03 de Mayo de 2015.] http://junit.org/.

24. **Chaffee, Alexander y Pietri, William.** Unit testing with mock objects. *IBM developerWorks.* [En línea] IBM, 01 de November de 2002. [Citado el: 04 de Mayo de 2015.] http://www.ibm.com/developerworks/library/j-mocktest/.

25. **Xamarin.** Create native iOS, Android, Mac and Windows apps in C#. *Xamarin.* [En línea] 2015. [Citado el: 14 de Mayo de 2015.] http://xamarin.com/platform.

8 Anexos

En esta sección se incluye información complementaria que es difícil de incluir en el cuerpo del documento debido a su tamaño o contexto no definido.

8.1 Ejemplo NUnit test

En la imagen siguiente se pude observar un fragmento de código correspondiente a uno de los test unitarios incluidos en el proyecto.

Para utilizar NUnit, se deben indicar una serie de directivas de compilación (`[TestFixture]`, `[Test]`) en los fuentes correspondientes. Estas directivas permite la serialización de los test. Como se puede en el fragmento de código, se utiliza un mock para simular el comportamiento de una clase de servicio de Django.

```
using System.Collections.Generic;
using Django.Core.DataAccess.Model;
using Django.MSM.Services.Impl;
using Django.MSM.UnitTest.mocks;
using NUnit.Framework;

namespace Django.MSM.UnitTest
{
  [TestFixture]
  public class AssetCustomServiceTest
  {
    [Test]
    public void GetAssetsByIdentifierPropertiesTest()
    {
        Asset assetTest = new Asset {ElectronicId = "TAG1"};

        AssetServiceMock assetServiceMock = new AssetServiceMock();
        assetServiceMock.Asset = assetTest;

        AssetCustomService.Instance.AssetServiceClass = assetServiceMock;

        string provider = "", year = "", model = "", serial = "";

        List<Asset> assets =
            AssetCustomService.Instance.
              GetAssetsByIdentifierProperties(
                provider, year, model, serial);

        Assert.AreEqual(1, assets.Count, "Error. Asset count");
        Assert.AreEqual("TAG1", assets[0].ElectronicId,
                        "Error. Asset count");
    }
  }
```

8.2 Planificaciones temporales

En este apartado se localizan las tablas de tiempo/esfuerzo que se corresponden con las planificaciones temporales realizadas durante el proyecto.

8.2.1 Planificación de modificaciones iniciales.

Las siguientes tablas se corresponden con las planificaciones temporales realizadas para las modificaciones preliminares en la aplicación Django MSM v1.0

Athelia

Quotation	MSM - Cambio de tag ARIO por ISO y declaración del Serial Number
Technical Budget	David Gonzalez
Request Date	28/11/2014
Quotation Date	01/12/2014

SumUp	PM	SWA / SWE	Description
Development			
Pure development		54 Hrs	See WBS tab
Versioning & Packaging		8 Hrs	Crear paquetes y comprobar que instala, hace upgrade, etc
Project Management			
PMI-based follow-up			
Software Quality Assurance			
Test Plan Document		EXCLUDED	Test plan document with all usecases defined and the tests for each use
Test Plan Execution		16 Hrs	Pruebas no documentadas de todas las operaciones
Documentation			
General Design			
User Manual		8 Hrs	Actualizar el manual de MSM y consensuar el cambio con el cliente
Delivery Dossier		2 Hrs	Smart Integration instructions provided to Athelia Customer Services
Detail Design			
Interface Design			
Deployment			
Installation/Configuration			
Integration Tests			
Training		8 Hrs	Training al key-user (en las instalaciones de MSM)
Total effort:		**96 Hrs**	**96 Hrs**
Total budget:	[redacted] €	[redacted] €	[redacted] €

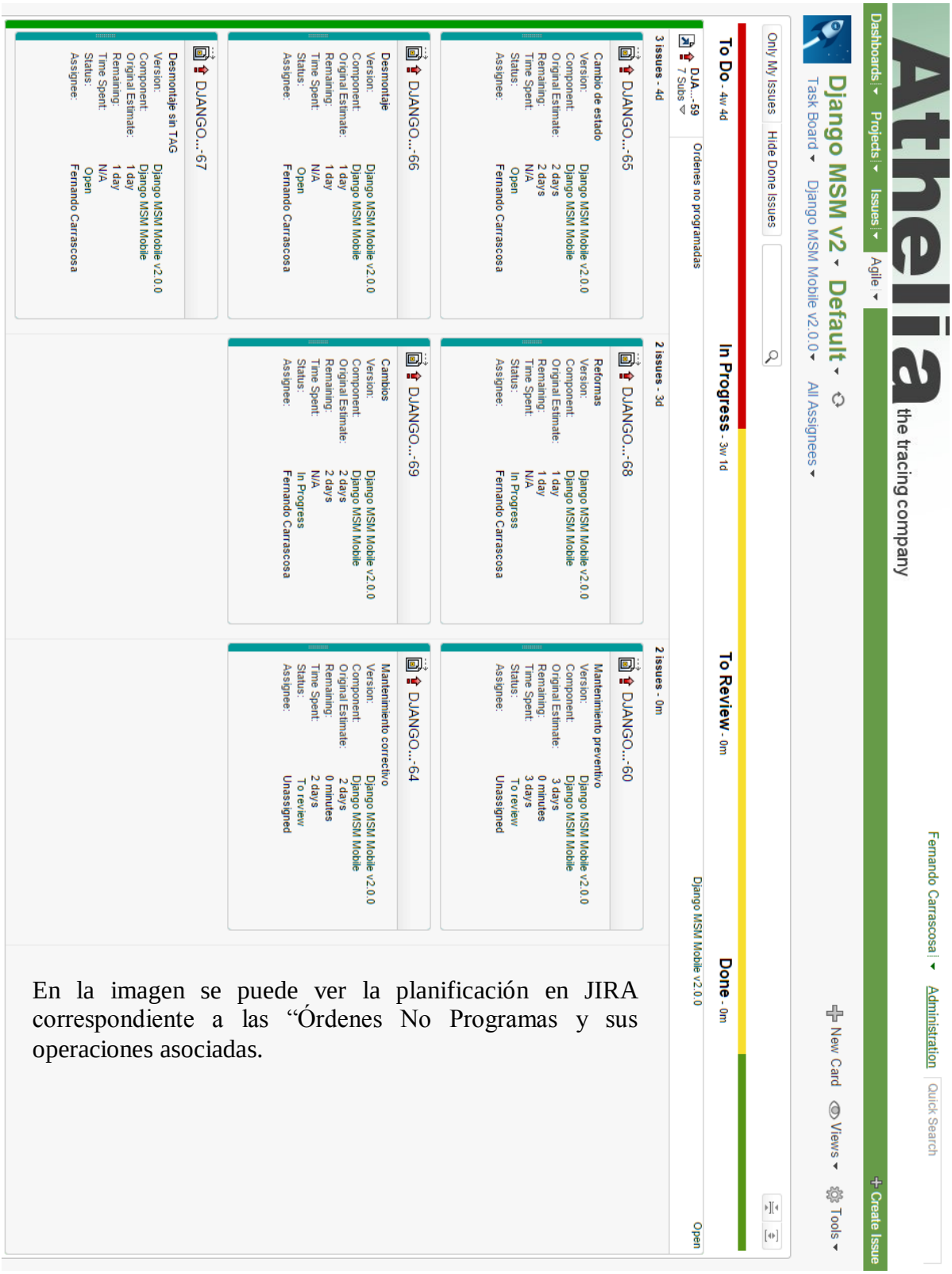
Ilustración 43: A.1 Quotation Update MSM v1.0

Código de seguimiento	Tarea o ítem	PV
1. Cambio de tag Ario por ISO y Asignación de Número de Serie		
1.1. HHC		
1.1.1	Duplicar la task de lectura de tag para crear una "extended" en la cual: > si el tag existe pero es un Ario, se obliga al usuario a cambiar por un ISO (que no figure en ningun otro asset del sistema) y posteriormente capturar el número de serie; > si el tag no existe, da un error; > si el tag existe, es un ISO pero no tiene número de serie, se lo pide al usuario; > si el tag existe, es ISO y dispone de S/N, continuar. La tarea dispara una WorkOrder de cambio de tag, y luego vuelve al flujo principal de la operación que se estuviese realizando en ese momento (correctivo, limpieza, etc)	16 Hrs
1.1.2	Nueva operativa en el menú principal para la matriculación de assets, que empieza con una lectura de tag: > si es un ARIO y NO existe, error > si es un ARIO y existe, exigir lectura de nuevo tag ISO, S/N > si es un ISO y NO existe, solicitar grupo del activo (portátil, fijo u otros), el tipo de activo, y el S/N > si es un ISO y existe, completar el S/N (si ya existe, se muestra pre-rellenado pero lo puede editar) Esta tarea dispara una WorkOrder de cambio de tag y S/N	12 Hrs
1.1.3	Modificar Django Core MSM para incluir el "save & send"	8 Hrs
1.1.4	Meter el auto-installer nuevo	8 Hrs
1.2. Integración de Datos		
1.2.1	Modificar la integración de los InstallationAsset de JDE para no pisar el S/N manejado internamente por Django	2 Hrs
1.2.2	Nueva integración de WO de matriculación o cambio de tag y S/N	8 Hrs
	TOTAL esfuerzo de desarrollo (sin PM & SQA, via ratios):	**54h**

Ilustración 44: A.1 WBS Update MSM v1.0

8.2.2 Planificación de tareas en JIRA

Detalles de la planificación a fecha 27 de Abril de 2015



En la imagen se puede ver la planificación en JIRA correspondiente a las "Órdenes No Programas y sus operaciones asociadas.

Ilustración 45: A2 JIRA Task Board 1

En la siguiente imagen se puede ver en detalle la planificación para el resto de operaciones de la aplicación. Esto incluye los distintos menús, el filtro de búsqueda, la matriculación de activos y las pruebas.

Ilustración 46: A2 JIRA Task Board 2

8.3 Fichero NAnt Django.MSM.build

A continuación se presenta la estructura general del documento Django.MSM.build que utilizará el software Jenkins para realizar las tareas de integración continua. Solo aparece en la imagen la estructura general, los procedimientos que se llevan a cabo dentro de las directivas `<call target="" />` son privados.

```
<?xml version="1.0"?>
<project name="Django.DAMM" default="build">
  <property name="nant.settings.currentframework" value="net-2.0"/>
  <!-- ================================================================== -->
  <!-- =                          main entry point                      = -->
  <!-- ================================================================== -->
  <target name="build">
    <!-- say to nant to call this target if something goes wrong -->
    <property name="nant.onfailure" value="build.failed"/>

    <!-- initializes the global variables -->
    <echo message="1. initializaing global variables" />
    <call target="initialization" />

    <!-- build the solution -->
    <echo message="2. binaries generation" />
    <call target="generation" />

    <!-- build the solution -->
    <echo message="3. version generation" />
    <call target="version" />

    <!-- Unit Test and code coverage -->
    <echo message="4. running unit/integration tests" />
    <!--<call target="test"/>-->
    <!-- Disabled until we fix an SQLite Interop version problem -->

    <!-- create the distribution package -->
    <echo message="5. distribution packaging" />
    <call target="package" />

    <!--Confluence Upload -->
    <echo message="6. Confluence Upload"/>
    <call target="confluence"/>

    <!-- tag release version -->
    <echo message="7. Tag release version" />
    <call target="tag" />

    <!-- Mail -->
    <echo message="8. Mail" />
    <call target="mail" />

  </target>
```